N° 70 mensuel Mai 1991-24 F

# La Revue de l'Utilisateur PC

*Le magazine professionnel en programmation PC*

**LES VIRUS DE PARTITION EDV, OHIO et STONED**

**XMS 2.0**

**INTERFAÇAGE**

**UTILISATION DES INTERRUPTIONS HVD, ASM**

**LE NOUVEAU CLIPPER 5.0 est arrivé**

**LE HASH-CODE démystifié**

MAI 1991 - N° 70 - 24 F - Suisse 7 FS - Belgique 190 FB - Canada 5,25 $ Can.

M 2537 - 70 - 24,00 F

LOGICIELS DERNIERES VERSIONS
POUR PC, XT, AT, COMPATIBLES,
UNIX, XENIX ET MACINTOSH AUX

**MEILLEURS PRIX**

# SOFTWORLD

17, Avenue Emile Zola, 75015 Paris. (Métro Javel)
Téléphone : (1) 40 59 02 99 Fax : (1) 45 79 95 55

## Pour MACINTOSH

| BASES DE DONNEES | | Prix Public TTC | Notre prix TTC |
|---|---|---|---|
| **4ème Dimension** | f | ~~7353~~ | 5885 |
| **File Force** | f | ~~3321~~ | 2250 |
| FoxBase Mac Mono | æ | ~~6820~~ | 5550 |
| Omnis 5 *Blyth* | f | ~~7110~~ | 6399 |
| **PAO / DESSIN** | | | |
| Page Maker 4.0 *Aldus* | f | ~~9607~~ | 7250 |
| Canvas 2.0 *Deneba* | f | ~~4140~~ | 3490 |

### *MATHEMATICA*

Un outil intégral de calcul, de représentation graphique 3D et de programmation symbolique interactive de haut niveau pour Scientifiques, Mathématiciens. Version Mac ou autres systèmes
Version 2.0 Mac Standard Nous contacter
Version 2.0 Mac Avancé Nous contacter

| | | | |
|---|---|---|---|
| **TABLEURS** | | | |
| Excel 2.2 *Microsoft* | f | ~~4732~~ | 3395 |
| WINGZ Avec graphe 3D | f | ~~5325~~ | 4350 |
| **EDITEURS DE TEXTE** | | | |
| Word 4 *Microsoft* | f | ~~3546~~ | 2450 |
| ***Formulator*** *(Mathématiques)* | a | ~~1950~~ | 1490 |
| **INTEGRES** | | | |
| Works *Microsoft* | f | ~~2953~~ | 2250 |
| **LANGAGES** | | | |
| **Think C 4.0** | **a** | **~~2953~~** | **1695** |
| **Think Pascal 3.0** | **a** | **~~2953~~** | **1695** |
| Mac SchemeLisp+Tools. | a | nc | 4250 |
| Prolog 2.0 *Adv.A.I. Syst.* | a | ~~3530~~ | 2950 |
| Professional Programmer's Ext. | a | nc | 2995 |
| **Prototyper** (Générateur de Codes sources) | a | nc | 1995 |
| QuickBasic *Microsoft* | a | ~~1175~~ | 895 |
| Smalltalk V/Mac *Digitalk* | a | ~~3084~~ | 2195 |
| Turbo Pascal *Borland* | f | ~~1180~~ | 850 |
| Turbo Pascal DataBase | a | ~~1180~~ | 850 |
| Turbo Pascal Numeric | a | ~~1180~~ | 850 |
| Turbo Pascal Tutor *Borland* | a | ~~1180~~ | 850 |
| **ZBasic 5.0** *Zedcor* | a | ~~2022~~ | 1395 |
| **SCANNER ET O.C.R.** | | | |
| Omnipage (O.C.R.) | a | ~~11563~~ | 9750 |
| Scanner couleur *Microtek* | a | ~~29532~~ | 24900 |
| **UTILITAIRES** | | | |
| Mac Tools Deluxe *Central P.* | a | ~~1529~~ | 1095 |
| SAM 2.0 (Antivirus) | a | ~~1175~~ | 990 |
| Virtual 030 pour Mac 68030 | a | ~~2360~~ | 1895 |

### INGENIERIE

| | | | |
|---|---|---|---|
| ACNAP Analyseur circuits AC | a | nc | 4550 |
| DCNAP Analyseur circuits DC | a | nc | 2600 |
| MICRO-CAP III (PC seulement) | a | nc | 14950 |
| SPP Traitement des signaux | a | nc | 1950 |
| Systat 5.0 Statistiques-Graphiques | a | nc | 7950 |

| | | | |
|---|---|---|---|
| **MUSIQUE** | | | |
| MacRecorder *Farallon* | a | nc | 2150 |
| Professional Composer *Mark Uni.* | a | ~~6227~~ | 4950 |

a : Version en américain.
f : Version en français.
æ : en américain avec possibilité d'échange en français.
nc : non communiqué

**TOUS NOS PRIX SONT TTC**
**PRIX INDICATIF AU 2 FEVIER 1991**
**Extrait de notre catalogue**
**Catalogue gratuit sur demande**

## POUR PC COMPATIBLES

| ADA et Librairies | | Public TTC | Notre TTC |
|---|---|---|---|
| Janus ADA Compiler +Tlkt | a | ~~5871~~ | 4950 |
| JanusADACompiler 386+Tkt | a | ~~5871~~ | 4950 |
| **ASM et Librairies** | | | |
| 386 DOS Extender *Pharlap* | a | nc | 4950 |
| Btrieve 5.00C *Novell* | a | nc | 5950 |
| EMS Toolkit *Realtime H.* | a | nc | 1350 |
| Intel ASM-386 Macro ASM | a | nc | 6000 |
| Intel ASM-86 Soft.Dev.Kit | a | nc | 7500 |
| MASM 5.1 *Microsoft* | a | ~~1767~~ | 1350 |
| PCX Effect + Source *Genus* | a | nc | 1895 |
| PCX Prog's Tlkit+Code Srce | a | nc | 5995 |
| PCX Text + Source *Genus* | a | nc | 2895 |
| **BASIC et Librairies** | | | |
| Basic 7.1 *Microsoft* | a | ~~4732~~ | 3795 |
| Basic Develop. Tools *Sterli.* | a | nc | 1175 |
| Btrieve *5.0 Novell* | a | ~~2906~~ | 2550 |
| FABS PLUS *Computer C* | a | nc | 2350 |
| Graph Pak Pro *Crescent* | a | nc | 1773 |
| Logic Gem *Sterling Castle* | a | nc | 1150 |
| MicroHelp Toolbox | a | nc | 1195 |
| PDQ *Cresent* | a | nc | 1295 |
| Toolbox II *MicroHelp* | a | nc | 1250 |
| PCX Effects + Source *Genus* | a | nc | 1895 |
| PCX Prog's Tlkit+Code Srce | a | nc | 5995 |
| PCX Text + Source *Genus* | a | nc | 2895 |
| Sciences & Engin. Tools | a | nc | 1195 |
| Quick Basic 4.5 *Microsoft* | f | ~~1175~~ | 895 |
| QBase *Crescent* | a | nc | 1495 |
| QuickPak Pro QB 4.5 *Cres.* | a | nc | 1595 |
| QuickPak Scientific *Crescent* | a | nc | 995 |
| QuickScreen QB4.5 *Crescent* | a | nc | 995 |
| QB Vol. 6 *MicroHelp* | a | nc | 1295 |
| Scientific SubroutineLibrary | a | nc | 2250 |
| Stay-res + *MicroHelp* | a | nc | 1295 |
| Turbo Basic *Borland* | æ | ~~1185~~ | 825 |
| Turbo Basic Toolbox chaque | æ | ~~1180~~ | 825 |
| True Basic *True basic* | a | nc | 995 |
| ZBasic *Zedcor* | a | nc | 1195 |
| **C, C++ et Librairies** | | | |
| C Asynch Manager *Blaise* | a | nc | 1890 |
| C Compiler 6.0 *Microsoft* | a | ~~4495~~ | 3250 |
| C Tools Plus *Blaise Comput* | a | nc | 1495 |
| GFX Graphics Library | a | nc | 2135 |
| GFX Fonts & Menus Lib. | a | nc | 2135 |
| GFX Font Editor | a | nc | 949 |
| GFX Screen Dump | a | nc | 949 |
| GreenLeaf Data Windows | a | nc | 4250 |
| Halo *Media Cybernetics* | a | nc | 3930 |
| High C 386 *MetaWare* | a | nc | 8950 |
| Lattice C *Lattice* | a | ~~5812~~ | 3350 |
| Power Screen *Blaise Comput* | a | nc | 1495 |
| Quick C 2.5 *Microsoft* | a | ~~1175~~ | 895 |
| Super Functions *Greenleaf* | a | nc | 2950 |
| Turbo C Tools *Baise Comp* | a | ~~2123~~ | 1495 |
| Turbo C++ 1.0 *Borland* | f | ~~1773~~ | 1250 |
| Turbo C++ 1.0 Pro *Borland* | f | ~~3552~~ | 2350 |
| Turbo C++ RuntimeLibrary | a | ~~1773~~ | 1495 |
| Win++ *Blaise Computing* | a | nc | 2495 |
| Zortech C++ 2.1 *Zortech* | a | ~~1773~~ | 1450 |
| Zortech C++ 2.1 Developper | a | ~~4501~~ | 2995 |

| COBOL et Librairies | | Public TTC | Notre TTC |
|---|---|---|---|
| Cobol 4.0 *Microsoft* | a | ~~8290~~ | 5950 |
| Cobol / 2 *Microfocus* | a | nc | 18000 |
| Cobol spII (Librairie) *Flexus* | a | nc | 4795 |
| **DEBUGGEURS** | | | |
| Intel ASM-386 Soft.Dev. | a | nc | 7500 |
| MultiScope - Windows | a | nc | 3995 |
| Periscope IV (25 MHz) | a | nc | 25950 |
| View232 *Blaise Computing* | a | nc | 1890 |
| **EDITEURS** | | | |
| Brief 3.0 *Solution Systems* | æ | ~~4495~~ | 2250 |
| EC Editor *C Source* | a | nc | 895 |
| Epsilon *Lugaru Software* | a | nc | 1950 |
| Norton Editor *Peter Norton* | æ | nc | 750 |
| VEdit Plus *CompuView* | a | nc | 1850 |
| **GENERATEURS** | | | |
| Automated Programmer*KGK* | a | nc | 4950 |
| Matrix Layout pr Basic, C, Pascal | a | nc | 1950 |
| PCYACC Corporate*Abraxas* | a | nc | 5500 |
| **INTELLIGENCE ART.** | | | |
| PC SCHEME *Texas Instru.* | a | nc | 1195 |
| Smalltalk V *Digitalk* | a | nc | 895 |
| Smalltalk V/286 *Digitalk* | a | nc | 1450 |
| Communication Pak *Digitalk* | a | nc | 425 |
| EGA/VGA Pak *Digitalk* | a | nc | 425 |
| Goodies 1 ou 2 ou 3 *Digitalk* | a | nc | 425 |
| µuLisp 90 *Soft Warehouse* | a | nc | 3558 |
| µuLisp+Compiler *Soft Ware.* | æ | nc | 4744 |
| **PASCAL/LIBRAIRIES** | | | |
| **MTASK pro *Ramsi*** | **f** | **~~2360~~** | **1950** |
| Pascal 4.0 *Microsoft* | a | ~~4732~~ | 3695 |
| Asynch Manager *Blaise C.* | a | nc | 1750 |
| Power Screen *Blaise C.* | a | nc | 1490 |
| Pascal View 232 *Blaise C.* | a | nc | 1890 |
| **Quick Pascal *Microsoft*** | **f** | **~~1175~~** | **895** |
| **Turbo Pascal 6.0 *Borland*** | **f** | **~~1773~~** | **1250** |
| **Turbo Pascal Pro 6.0** | **f** | **~~3552~~** | **2350** |
| **AUTRES LANGAGES** | | | |
| Actor *White Water Group* | a | ~~10615~~ | 7950 |
| Fortran 5.0 *Microsoft* | a | ~~5990~~ | 4250 |
| Guideline C++ *Guidelines* | a | nc | 3950 |
| **DIVERS** | | | |
| Dr DOS 5.0 | f | ~~1174~~ | 950 |
| Merge 386 2 Utilisateurs *Locus* | a | ~~4151~~ | 3995 |
| Nantucket Clipper 5.0 | f | ~~11800~~ | 9250 |
| Norton Guides *Norton* Chque | a | ~~1186~~ | 995 |
| PC MOS/386 Mono *Soft.Link* | a | ~~2953~~ | 2495 |
| Window 3.0 *Microsoft* | f | ~~2360~~ | 1650 |
| VM 386 *IGS* | f | ~~3499~~ | 3050 |
| VM 386 Net Pak *IGS* | f | ~~2965~~ | 2650 |
| Windows Devel. Toolkit 3 | a | ~~4495~~ | 3295 |

**UNIX**
Systèmes / Langages / Fichiers / SGBD
Editeurs de Texte / Utilitaires
**Nous contacter : (1) 40 59 02 99**

| BASES DE DONNEES | | Public TTC | Notre TTC |
|---|---|---|---|
| dB XL *Wordtech systems* | f | ~~4715~~ | 3850 |
| Fox Base+ 2.10 *Fox Soft.* | f | ~~9429~~ | 7500 |
| Fox Pro Mono *Fox Soft.* | f | ~~10615~~ | 8150 |
| Genifer+Template CompilerFoxBas | a | nc | 4195 |
| Paradox 3.5 *Borland* | f | ~~9963~~ | 7150 |
| Super DB *Computer Asso.* | f | ~~7021~~ | 5200 |
| Super Base 4/Windows | f | ~~7412~~ | 5950 |
| **CAO / DAO / PAO** | | | |
| Design CAD 2D (3D +1000 Frs) | f | ~~4685~~ | 3950 |
| Page Maker 4.0 PC *Aldus* | f | ~~9610~~ | 6750 |
| Ventura 2.0 *Rank Xerox* | f | ~~9192~~ | 8550 |
| **COMPTA/GESTION** | | | |
| Ciel! Compta-Gestion *Ciel!* | f | ~~1156~~ | 925 |
| Compta Saari Major *Saari* | f | ~~16905~~ | 13950 |
| **INTEGRES** | | | |
| Ability Plus Dual *Migent* | f | ~~1927~~ | 1695 |
| Works PC 2.0 *Microsoft* | f | ~~2950~~ | 1995 |
| **TABLEURS** | | | |
| Compile 12C Compiler WK. | a | nc | 8950 |
| Excel PC (286 et 386) | f | ~~5918~~ | 3895 |
| Lotus 123 Version 3.0 | a | ~~6750~~ | 4950 |
| Multiplan 4 *Microsoft* | f | ~~3309~~ | 2395 |
| Quattro Pro 2.0 *Borland* | f | ~~5924~~ | 4150 |
| VP Planer + 3D *Paperback* | f | ~~3440~~ | 2595 |
| **TRAITEMENT/ TEXTE** | | | |
| Chiwriter (Scientifiques) | f | ~~1235~~ | 1195 |
| **MathEdit (Mathématiques)** | a | ~~2366~~ | **2366** |
| Sprint 1.5 *Borland* | f | ~~2959~~ | 2050 |
| Word 5.5 *Microsoft* | | ~~5325~~ | 3550 |
| Word pour Window *Micros.* | f | ~~5918~~ | 4150 |
| Wordperfect 5.1 *Wordperf.* | f | ~~5693~~ | 4595 |
| **UTILITAIRES / SYST.** | | | |
| Above Disc EMS 4.0 | æ | ~~1510~~ | 1195 |
| CopyWrite *Quaid* | a | ~~1175~~ | 695 |
| Disk Explorer *Quaid* | a | ~~1175~~ | 695 |
| Norton Adv. Utilities 5.0 | a | ~~1779~~ | 1450 |
| Norton AntiVirus | f | ~~1175~~ | 950 |
| **PC Tools Deluxe 6.0** | **f** | **~~2005~~** | **1395** |
| **SOURCER+BIOS** | **a** | **~~2075~~** | **1550** |
| **MATHEMATIQUES** | | | |
| **Derive 2.01** *Soft Warehouse* | **a** | **~~2965~~** | **1928** |
| Greenleaf Business MathLib | a | nc | 3950 |
| Grapher *(Graphiques scientif.)* | a | ~~2953~~ | 2366 |
| HiWire Plus Wintek | a | nc | 9500 |
| MathCAD 2.5 *MathSoft* | a | ~~6274~~ | 4950 |
| Mathematica DOS-386 V2.0 | a | Tel | Tel |
| Mathematica DOS-386/7 V2 | a | Tel | Tel |
| Mathematica DOS-386/Weitek V2 | a | Tel | Tel |
| QuickPak Scientific *Crescent* | a | nc | 995 |
| Sciences & Engineer. Tools | a | nc | 1195 |
| Surfer *(Surface, carte géo. 3D)* | a | ~~6111~~ | 4950 |
| SYSTAT 5.0 + SYGRAPH | a | nc | 8995 |
| TKSolver | a | nc | 4685 |

### PRO-CITE

**Gestion bibliographique. Version US pour Compatibles ou Macintosh**
~~4684,70 FF TTC~~ **PROMO 3950 FF TTC**

## BON DE COMMANDE
## SOFTWORLD

**17, Av. Emile Zola, 75015 Paris. ( Métro Javel )**
**Téléphone : (1) 40 59 02 99 / Fax : (1) 45 79 95 55**

Mon ordinateur est du type : (Macintosh) (PC Compatible)
Format de disquette : 3 1/2 ○ ou 5 1/4 ○
Je veux commander les produits notés ci-contre.
Mode de paiement :
○ Par Chèque ci-joint.
○ Par Contre remboursement +40 Frs (montant d'achat <5000 Frs)
○ Par Carte Bleue N° _____ Date d'expiration : _____
Date :
Signature : 1991

Nom : _____ Prénom : _____
Adresse : _____
_____
Code postal : _____ Localité : _____
Téléphone : _____

| Désignation | Quantité | Prix TTC |
|---|---|---|
| | | |
| | | |
| Frais d'envoi : | | |
| Total TTC : | | |

**Frais d'envoi recommandé obligatoire 40 Frs par 1000 Frs**

**Aucun règlement ne sera encaissé avant l'expédition des marchandises**

Nous fournissons la plupart des produits français et étrangers à des prix défiant toute concurrence et dans un délai record.
Plus de 3000 produits pour PC et MAC.
Applications verticales, Bases de données, D.A.O., Editeurs de Textes, Graphiques, Ingénierie, Intégrés, Langages, Mathématiques, Librairies, P.A.O, Statistiques, Tableurs, Utilitaires...

**Heures d'ouverture**
**10h à 12h et de 14h à 19h**
**Fermé Samedi , Dimanche et Fériés**

La Revue de l'Utilisateur PC

Le magazine professionnel en programmation PC

Mensuel 24 F
2, rue d'Amsterdam
75009 Paris
Tél. : (1) 42.81.54.27.
Service abonnement :
Tél. : (1) 42.81.54.27.

Directrice de la Publication :
Sardie ERRERA

Editeur :
Roger CHRISTOPHE

Rédacteur en chef :
Paul LAMBASLE

Directeur de fabrication :
Michel MILLET

Directeur de publicité :
Liliane ERRERA
Tél. : (1) 42.67.93.80.

Edité par SOFTin Communications (France) Sarl
R.C. Paris B 329 834 592
N° Siret
329 834 592 000 19
Commission Paritaire (en cours)
Dépôt légal :
2e trimestre 1991

CCP Paris 8850 84 D

Impression :
SNIL
93600 Aulnay-sous-Bois

Composition :
SOFTin Communications

2

**UTILISATION DES INTERRUPTIONS HVD, ASM**

Exploiter les possibilités d'une machine après sa mise en route, le système d'exploitation
Par C. LAMBEY

16

**NANTUCKET .5**

La version française de Clipper 5.0 est arrivé
Par Dominique BRUYERE

28

**VIRUS SUITE LES VIRUS DE PARTITION EDV, OHIO ET STONED**

Une mécanique pour venir à bout avec les virus de partition avec le TURBO C++
Par Jean-Luc NAIL

44

**XMS 2.0**

Ou les secrets de HIMEM.SYS de Microsoft. XMS étant le sigle de Extended Memory Specification
Par Daniel VASSEUR

50

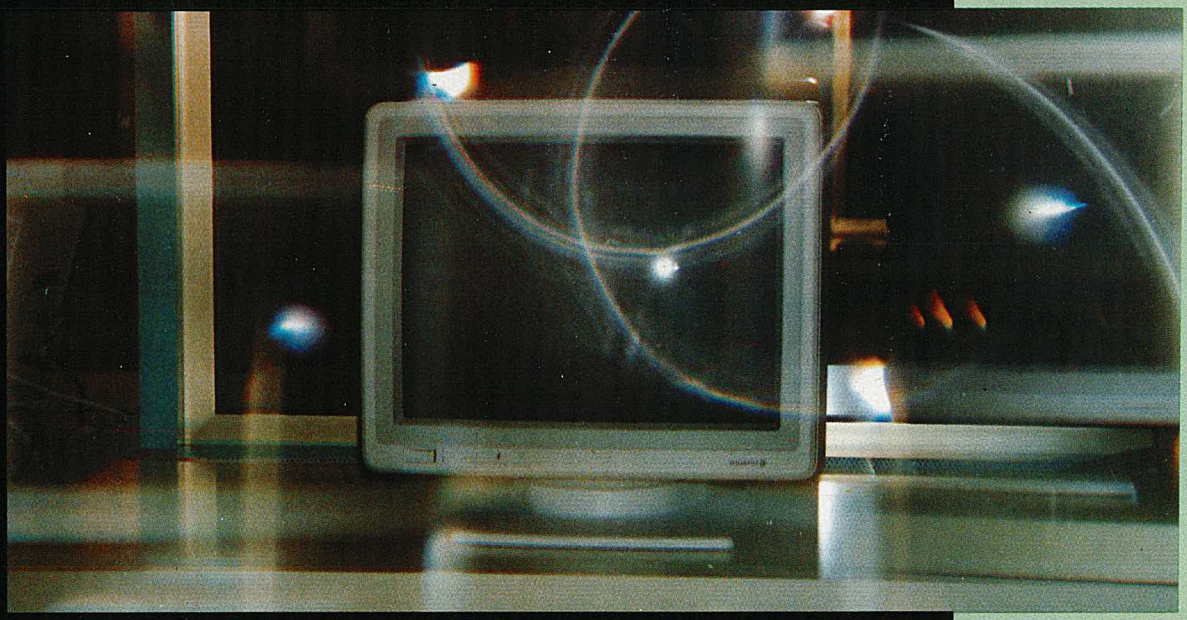
**INTERFAÇAGE**

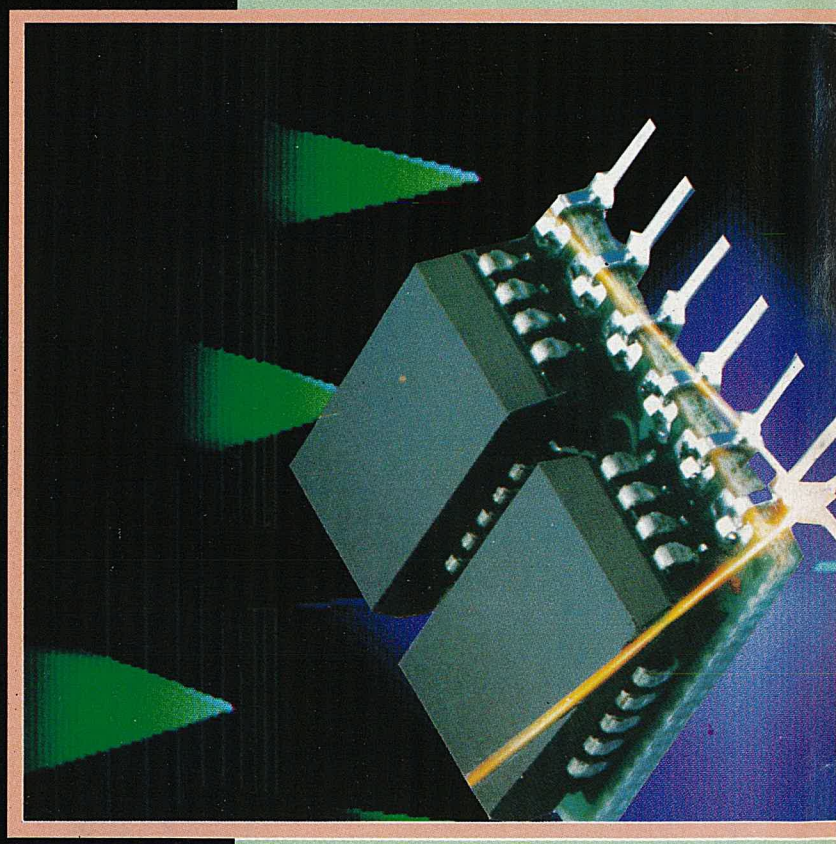
Le but de cet article est d'expliquer comment s'effectue le transfert de données à des sous-programmes et comment on peut interfacer différents langages
Par Jean-Charles LE TARNEC et Magali S.

58

**LE HASH-CODE DEMYSTIFIE**

Parfois les programmes utilisent des techniques rustiques pour stocker les données
Par Yves ROUMAZEILLES



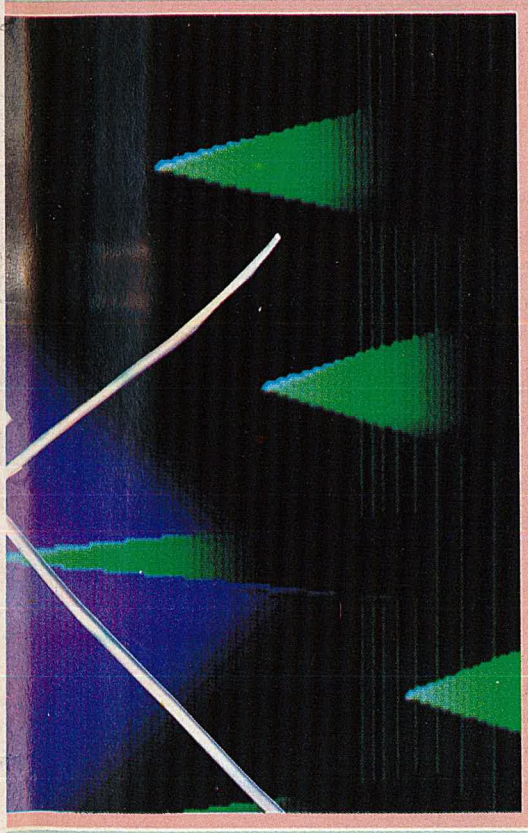


**Exploiter les possibilités d'une machine après sa mise en route, c'est avoir recours à un serviteur muet dont personne ne sera surpris qu'on lui ait donné le nom de Système d'Exploitation !**

Bernard C. LAMBEY

# Utilisation des Interruptions HVD.ASM

Et, tel les poupées russes qui s'emboîtent les unes dans les autres, ce serviteur muet, le Processeur de commande



("Shell", enveloppe extérieure, "command.com") dispose, dans la poche de son gilet d'un autre serviteur muet, le DOS — Disk Operating System — qui, lui est en général plus connu, au moins de nom.

Mais ce n'est pas fini ! Car, à son tour, le DOS cache sous son trousseau de clés un petit valet infatigable, ultra rapide, hyper efficace... et bavard ! C'est lui qui a reçu le nom de BIOS.

Le "BIOS" — Basic Input Output System — d'un compatible est logé dans une ROM de la machine et recopié en RAM à la mise sous-tension. Il est constitué, en autres, d'un ensemble de routines soigneusement optimisées.

Ces routines portent le nom d'INTERRUPTIONS parce qu'elles sont conçues pour disposer d'une priorité de traitement maximum. Elles interrompent donc l'exécution du programme qui les appelle, prennent la main, traitent le problème particulier dont elles sont spécialistes, puis rendent la main (avec ou sans passage de paramètres) au programme appelant, lequel peut reprendre son cours normal.

Tous les traitement de base des entrées et des sorties de la machine sont ainsi sous-traitées par les routines du BIOS, que les périphériques concernés soient le clavier, l'écran, le modem, l'imprimante ou les périphériques de mémorisation, disques et disquettes, bandes etc...

Et cela va réduire de plus des deux tiers la longueur des programmes. Pas moins ! Cela paraît incroyable, mais ceux qui ont la chance de disposer de logiciels d'examen du type CodeView (Microsoft) ont pu le constater. Lorsqu'on suit, pas à pas, le fonctionnement d'un petit programme, on constate que la machine passe plus de deux tiers de son temps à exécuter des instructions du BIOS, sur ordre du programme appelant.

Alors, pensez-y, que serait la taille de tel traitement de texte de notre connaissance, qui fait déjà quatre méga-octets, s'il n'avait pas la possibilité de recourir aux interruptions !...

Les avantages de la méthode sont importants. Si le BIOS de la machine a été bien conçu, chaque appui sur une touche, chaque affichage, chaque appel d'un fichier sera traité avec le maximum d'efficacité et de rapidité, et l'utilisateur sera satisfait !

En fait, quatre vingt pour cent des utilisateurs ne vont pas plus loin. Et tels les conducteurs de véhicules qui ne veulent pas savoir ce qu'est une soupape ou le train épicycloïdal de leur boîte de vitesse, du moment que "ça" marche, tout va bien !

Certains, cependant, aiment comprendre. C'est à ceux-là que s'adresse notre revue. Nous allons donc leur donner un exemple de recours aux interruptions.

Pour cela, le probleme que nous allons tenter de résoudre est un de ces petits problemes que l'utilisation informatique de la notation hexadécimale rend souvent irritant. Si le fait d'avoir dix doigts aux mains nous fait juger naturelle la notation décimale, l'hexa, lui, ne nous parle pas a l'esprit.

Et le mélange de lettres et de chiffres n'arrange rien. Il faut, lorsqu'un "FA3E" se présente, une grande habitude pour le "mentaliser" aux environs de 64062... Avouez que c'est vexant, alors que notre puissante machine est là, devant nous, de devoir rechercher la calculette (si elle sait le faire !) pour lui poser la question. Car il faut la retrouver, cette fichue calculette...

Nous allons donc élaborer un petit utilitaire qui se chargera de la conversion d'un nombre hexadécimal en décimal, et pendant que nous y sommes, nous lui demanderons aussi de nous donner la correspondance en binaire, puisque c'est ce qu'utilise la machine (vous savez, le fameux "courant = 1", pas courant = 0 !).

Nous aurons besoin pour le réaliser en assembleur de trois interruptions.

— Interruption 16h : c'est une des routines que déclenche la frappe d'une touche au clavier. Elle est chargée de lire trois indications relatives à la touche frappée, dont deux seulement nous seront utiles : le scan-code et le code Ascii. Le scan-code, c'est le nom de la touche, et le code Ascii sa valeur.

On conçoit donc que pour un nom de touche on puisse décider de plusieurs valeurs. C'est bien ce qui se passe. La touche de scan-code 16 (dec) peut avoir, selon le cas, les valeurs 97 (a), 65 (A), 01 (Ctrl + a) etc...

— Interruption 10h : c'est l'interruption vidéo. Elle a de multiples fonctions. Elle se charge de transmettre à l'écran le ou les caractère(s) à afficher, en gérant, ou non, la place du curseur, ligne et colonne, les couleurs du papier et de l'encre, la page d'affichage, l'effacement, le scrolling, etc...

— Interruption 21h : on l'appelle "Interruption DOS" parce c'est la routine qui assure toutes les fonctions les

# LA GAMME "PEARL" D' ALT COMPUTER

## PS - 2010
- Processeur 8088 à 10 MHz
- 640 Ko de mémoire
- 5 slots d'extension
- 1 port série, 1 port parallèle et 1 port jeu
- Contrôleur disque et disquette
- 1 lecteur 360 Ko
- Clavier 102 touches
- Système d'exploitation DR-DOS
- Coffret SLIM. Alimentation 150 W

## PS - 3012 (PS - 3016)
- Processeur 80286 à 12 MHz (16 MHz)
- 1 Mo de mémoire, extensible à 4 Mo
- 7 slots d'extension
- 2 ports série, 1 port parallèle et 1 port jeu
- Contrôleur disque et disquette AT-IDE
- 1 lecteur 1.2 Mo ou 1.44 Mo
- Clavier 102 touches
- Système d'exploitation DR-DOS
- Coffret BABY. Alimentation 200 W

## PS - 4012 (PS - 4016)
- Identiques aux PS-3012 (PS-3016) avec coffret DESK-TOP

## PS - 6016 (PS - 6020)
- Processeur 80386SX à 16 MHz (20 MHz)
- 1 Mo de mémoire, extensible à 8 Mo
- 7 slots d'extension
- 2 ports série, 1 port parallèle et 1 port jeu
- Contrôleur disque et disquette AT-IDE
- 1 lecteur 1.2 Mo ou 1.44 Mo
- Clavier 102 touches
- Système d'exploitation DR-DOS
- Coffret Mini-Tower. Alimentation 200 W

**RECHERCHONS**

**DISTRIBUTEURS**

**et**

**REVENDEURS**

ALT COMPUTER se réserve le droit de modifier, sans préavis, les caractéristiques de ses produits.

## PS - 7025
- Processeur 80386 à 25 MHz
- 1 Mo de mémoire, extensible à 12 Mo
- 7 slots d'extension
- 2 ports série, 1 port parallèle et 1 port jeu
- Contrôleur disque et disquette AT-IDE
- 1 lecteur 1.2 Mo (5.25")
- Clavier 102 touches
- Système d'exploitation DR-DOS
- Coffret Tower. Alimentation 220 W

## PS - 70C25 (PS - 70C33)
- Identique au PS-7025 avec 64 Mo de mémoire cache; vitesse du processeur 25 ou 33 MHz

## PS - 80C25 (PS - 80C33)
- Processeur 80486 à 25 MHz (33 MHz)
- 1 Mo de mémoire, extensible à 16 Mo
- 128 Ko de mémoire cache
- 7 slots d'extension
- 2 ports série, 1 port parallèle et 1 port jeu
- Contrôleur disque et disquette AT-IDE
- 1 lecteur 1.2 Mo (5.25")
- Clavier 102 touches
- Système d'exploitation DR-DOS
- Coffret Tower. Alimentation 220 W

## SCS - 3000
- Portable "Note Book"
- Processeur 80386SX à 20 MHz
- 1 Mo de mémoire, extensible à 4 Mo
- Ecran LCD VGA (480x640)
- 2 ports série, 1 port parallèle
- 1 disque dur 20 Mo
- 1 lecteur 1.44 Mo (3.5")
- Clavier 80 touches
- Batterie rechargeable
- Système d'exploitation DR-DOS
- Poids 3.5 Kg

Ces configurations sont livrées sur demande avec :

- Ecran 14" monochrome ou VGA couleur 1024 x 768 - Pitch = 0.28
- Disques durs de 40 Mo à 1 Go; contrôleurs AT-IDE, SCSI, ESDI.

**PS - 4016**

Processeur 80286 à 16 MHz

**PS - 2010**

Processeur 8088 à 10 MHz

**PS - 3012**

Processeur 80286 à 12 MHz

**SCS - 3000**

Processeur 80386SX à 20 MHz
ECRAN VGA (3.5 Kg)

**PS - 6016**

Processeur 80386SX
à 16 ou 20 MHz

**PS - 70C25**

Processeur 80386
à 25 ou 33 MHz

**PS - 80C25**

Processeur 80486
à 25 ou 33 MHz


NLT COMPUTER

1, Avenue de Laponie - Z.A. de Courtabœuf - 91951 LES ULIS CEDEX FRANCE
Tél. : 33 (1) 64.46.68.63 - Télex : 603 261 F ALTCHNO - Téléfax : 64.46.66.50

plus courantes du DOS, bien que celui-ci propose de nombreuses autres interruptions aux utilisateurs. C'est un ensemble complexe, avec fonctions et sous-fonctions, dans lequel nous n'entrerons pas, sauf pour signaler que tout programme a recours à elle, ne serait-ce que pour signaler à la machine qu'il est terminé !... Ici, nous l'utiliserons pour des affichages de chaînes de caractères.

Toutes les interruptions ont en commun deux points.

1) Leur commande d'appel est la même : INT suivi du numéro de l'interruption en hexadécimal. On fait toujours suivre ce numéro d'un "h" à la lecture duquel l'assembleur sait que le nombre en question est à lire en hexadécimal. Un "b" aurait indiqué du binaire, et l'absence de lettre : du décimal.

2) Beaucoup d'interruptions sont capables d'actions diverses, voire inverses. Il leur faut donc une indication du traitement désiré. Ce traitement est toujours signalé à la machine par un nombre entré dans le registre AH. C'est le code de fonction ou de service.

Cela nécessite souvent la protection des données que AH pourrait héberger avant entrée de ce code de traitement, faute de quoi la précédente donnée serait perdue à jamais.

## Interruptions et Fonctions utilisées dans HVD.ASM

Int 10h - Service 2 : met le curseur à une place déterminée; a besoin de la page écran dans BH, de la ligne désirée dans DH, de la colonne dans DL. L'interruption ne "donne" rien. Le curseur saute en xy.

Int 10h - Service 3 : indique la position du curseur ; a besoin de la page courante dans BH. Donne la position du curseur : DH contient la ligne (y), DL contient la colonne (x), CH contient ligne début curseur, CL contient ligne fin curseur.

L'identité des registres utilisés par ces deux fonctions n'est pas le fruit du hasard ! Elle permet d'utiliser les deux fonctions l'une derrière l'autre sans réintroduction de données. Ce sera le cas ici.

Int 10h - Service 6 : scrolling d'une fenêtre à déterminer. Deux sous-fonctions : si AL=0, effacement de l'écran ; si AL<>0, c'est le nombre de lignes à dérouler vers le haut. Caractéristique de la fenêtre : BH, attribut couleur, papier/encre ; CH, ligne coin haut gauche ; CL, colonne coin supérieur gauche ; DH, ligne coin inférieur droit ; DL, colonne coin inférieur droit. L'interruption exécute l'opération souhaitée et ne "donne" rien en sortie.

Int 10h - Service 8 : lecture caractère et attribut à la position du curseur ; a besoin de la page écran dans BH. En sortie donne : AL, contient le caractère lu ; AH, contient l'attribut du car.

Int 10h - Service 9 : écriture d'un caractère (sans gestion du curseur) ; a besoin de la page écran dans BH, de l'attribut couleur en BL, du caractère à afficher en AL. L'interruption ne "donne" rien. L'affichage a lieu à la position curseur.

Int 10h - Servicce 0Eh : affiche le car logé dans AL ; gère le curseur, le scrolling et le CR ; a besoin du caractère dans AH, du numéro de page dans BH. L'interruption ne "donne" rien, mais affiche le caractère.

Int 10h - Service 0Fh : lecture de l'état courant vidéo ; n'a besoin de rien à l'entrée. En sortie donne : AL, contient le mode courant ; AH, contient Nb caractères/ligne ; BH, contient numéro de page active.

Int 16h - Service 0 : lecture (avec attente) d'un caractère entré au clavier ; n'a besoin d'aucun paramètre d'entrée. En sortie, fournit deux éléments : AH, contient à présent le scan-code ; AL, contient le code Ascii qui dépend de l'état clavier (Shift ou non).

Int 21h - Service 2 : envoi à l'écran d'un caractère ; a besoin du code Ascii du caractère dans DL. Rien en sortie.

DANS LA JUNGLE DES LANGAGES DE DEVELOPPEMENT, VOICI DES RAISONS DE FAIRE LE BON CHOIX

TURBO-FORTH

c'est:

- un **interpréteur** et un **compilateur** pour IBM PC/XT/AT/PS et compatible,
- la définition **d'objets nouveaux**,
- un débogueur-traceur,
- un **décompilateur**,
- un **assembleur** utilisant de véritables mnémoniques et l'assemblage structuré,
- l'appel aux programmes externes, y compris ceux écrits en C ou PASCAL,
- le traitement hiérarchisé des fichiers source ASCII et **l'interprétation ré-entrante** de chaînes alphanumériques.

Des utilitaires: virgule flottante CORDIC, **méta-compilateur**, accès aux ports, accès aux fichiers dBASE, **exécution multi-tâche**, etc...

**TURBO-FORTH**, produit **français**, offre près de 1 Mo de fichiers sources et exécutables. Il est accompagné de deux manuels (Manuel d'apprentissage et Guide de référence). Il fonctionne sous DOS 2.0 et suivants. Configuration minimale requise: 128 Ko RAM...!!!
**TURBO-FORTH** convient particulièrement aux applications 'temps-réel' nécessitant un encombrement mémoire réduit (applications résidentes), automatismes, régulation, analyse de signaux, transmission, traitement de données complexes, graphisme, calculs scientifiques, programmation et simulation de circuits logiques, cross-compilation, transcodage de données, cryptage/décryptage, gestion de bases de données...

**TURBO-FORTH 995,00 FF** (838,95 FF HT)
Forfait port et emballage 62,50 FF en sus

REM CORP - 17, rue de la Lancette - 75012 PARIS (France)
tel: (33) 1- 43.40.96.53 fax: (33) 1- 43.42.32.15
Minicom: 3612 REM CORP 143409653

Int 21h - Service 9 : envoi à l'écran d'une chaîne de caractères ; a besoin de l'adresse de chaîne dans DX. La chaîne doit se terminer par un "$". Pas de possibilité de gérer les couleurs de l'affichage (seule Int 10h en est capable). (Le segment dans DS est supposé être le segment courant).

Int 21h - Service 04Ch : fin de programme ; n'a besoin de rien d'autre. En sortie : AL, contient un code de sortie exploitable par "if errorlevel" dans un éventuel batch de lancement.

Note 1 : Quand deux registres huit bits (DH et DL par exemple) sont utilisés pour des entrées, on peut les coupler si l'on a pris la précaution de convertir en hexadécimal les données à y charger :

mov DH, 5 et mov DL,53 peuvent être remplacés par mov DX,0535h

puisque le registre 16 bits DX correspond à DH-DL.

Note 2 : La macro MUL10h est écrite pour se substituer à l'instruction MUL qui est la plus lente de l'assembleur 8086. Pour une multiplication par 16 (10h) MUL10h est environ soixante-dix fois plus rapide que MUL. En partant d'elle on peut écrire des macros multipliant par (nx2) sans grande difficulté.

## Le programme HVD.ASM

(HVD pour "Hexa Vers Décimal").

Disposant des outils, on va pouvoir écrire le programme.

1) On va établir une liste des scan-codes des touches qui correspondent aux chiffres et aux lettres utilisés en hexadécimal, soit de 0 à 9 et de A à F. Puis on dresse une seconde liste, celle des codes Ascii des mêmes chiffres.

La procédure LEBON examine la première et nous donnera la valeur de la touche, prise à la même position dans la seconde. Ceci évitera que l'on ait à se mettre en majuscules bloquées pour utiliser le programme, lequel fonctionnera aussi bien en majuscules qu'en minuscules. On se limitera volontairement au clavier alphanumérique, puisqu'au demeurant celui-ci nous est imposé par la frappe des lettres.

2) Les codes Ascii obtenus seront mémorisés dans une chaîne de caractères où on les placera au rang idoine, selon le nombre de chiffres (digits) constituant le nombre à convertir. Cela permettra, entre autres, d'obtenir un formatage du binaire plus facilement lisible.

3) On ira les chercher.

— Une première fois pour un affichage de contrôle.

— Une deuxième fois pour la conversion du nombre hexadécimal en nombre décimal, et ceci en deux étapes.

— Etape 1 : transformation des codes Ascii en digits. La procédure DEDUIT assurera cette conversion.

La chaîne, dès ce moment n'est plus affichable, mais elle représente le nombre hexadécimal lui-même. Il nous servira pour l'étape suivante et pour l'affichage du binaire.

— Etape 2 : le nombre hexadécimal est "décimalisé" par une succession de soustractions avec affichage du résultat obtenu. C'est le rôle des deux procédures CONVERTIT et METDEC.

— On reprendra la chaîne une troisième fois pour l'affichage en binaire. L'hexadécimal est soumis 16 fois à une rotation vers la gauche de l'ensemble de ses 16 bits. Le flag carry est mis à 1 si le bit est 1, et 0 si le bit est nul. Il suffit alors d'écrire l'état de ce flag en lui ajoutant le code Ascii de 0 pour le rendre affichable.

Pour rendre l'affichage plus lisible, une alternance de couleurs sera utilisée, singulièrement pour l'affichage du binaire. Afin de faciliter la lecture des seize bits, ceux-ci sont découpés en deux groupes de huit, (Octet le plus signifiant, et octet le moins signifiant), dans lesquels les couleurs changent d'un bit à l'autre.

Le logiciel, d'emploi très simple, peut

se relancer lui-même. Des affichages de messages en permettant une utilisation facile.

Chaque ligne du source est munie d'un commentaire explicatif qui doit permettre une compréhension aisée du rôle de chaque ligne d'instructions.

Chaque ligne du source est munie d'un commentaire explicatif qui doit permettre une compréhension aisée du rôle de chaque ligne d'instructions.

## Fonctionnement

Le logiciel attend la frappe de touches correspondant aux seize digits hexadécimaux. Comme un hexadécimal dépassant FFFF ne peut être utilisé par le processeur, le programme se lance tout seul à l'entrée du 4e chiffre.

Un nombre de moins de 4 digits nécessitera la frappe de la touche RETURN pour lancer le programme.

Exemple de résultat à l'écran.

```
┌──────────────────────────────────────────────────┐
│        Conversion: Hexadécimal -> Décimal        │
├──────────────────────────────────────────────────┤
│Entrer un hexadécimal, sans shift (clavier alpha) │
│     Noir: Touche tapée, Cyan: Digit entré        │
├──────────────────────────────────────────────────┤
│              (C) Bernard C. Lambey - 1990        │
└──────────────────────────────────────────────────┘
          (Fin d'introduction par RETURN)
┌──────────┬──────────┬──────────┬──────────────────┐
│ Entrée   │ Mémorisé │ Décimal  │      Binaire     │
└──────────┴──────────┴──────────┴──────────────────┘
fFaA"3eE       FA3E       64062      11111010 00111110
aAbBcCdD       ABCD       43981      10101011 11001101
/1é2"3'4       1234       04660      00010010 00110100
fFfFfF          FFF       04095      00001111 11111111
fFfF             FF       00255      00000000 11111111
fF                F       00015      00000000 00001111
```

Dans la colonne ENTREE, chaque frappe d'une touche fait apparaître en noir la valeur usuelle de la touche, immédiatement suivie, en Cyan, du digit hexadécimal que la machine déclare avoir enregistré.

La colonne MEMORISE permet un contrôle de "conformité".

La colonne DECIMAL donne le nombre décimal demandé.

La colonne BINAIRE donne les seize bits du même nombre mais affiché, cette fois-ci, en base 2.

Toute autre frappe que celle correspondant à un digit hexadécimal est refusée avec émission du message d'erreur orienté vers une réintroduction. Mais à ce niveau une sortie par Escape est possible.

Sur la droite du tableau d'affichage une petite fenêtre redisposable en fonction du scrolling reçoit les deux invites propres au programme.


NOUVEAU

**NOUVEAU PRIX ! 4.500,00 F. H.T.**

- ☞ TRAVAILLE AVEC TOUS LES LANGAGES SOUS M.S. DOS®.
- ☞ REDUCTION DE 30% A 70% DE L'IMAGE MEMOIRE.
- ☞ SUPPORTE M.S. CODE VIEW®.
- ☞ PHASE DE LINKAGE RAPIDE.
- ☞ OVERLAYS DYNAMIQUES.
- ☞ "BOITE A OUTILS" SUR LES GESTIONS D'OVERLAYS.
- ☞ DEVELOPPEMENT POSSIBLE SOUS OS/2® ET SOUS WINDOWS®.
- ☞ PAS DE MODIFICATION DU CODE SOURCE.
- ☞ PAS DE ROYALTIES.
- ☞ MODE V.M.L, EXEMPLES:

VML

FILE MAIN, MENU, CALC.

FILE SUBMENU, REPORTS.

LIB SCREEN, DATABASE.

... et RTLink crée automatiquement les Overlays dynamiques dans votre programme.

... dans la plupart des cas, 3 ou 4 Mo de programmes tourneront dans 300 Ko de mémoire DOS conventionnelle.

.RTLink/Plus™

**Développeurs !!**

**Les programmes requierrent toujours plus de mémoire**

**Résolvez vos problèmes avec RTLink®!!**

**RELEASE 4.0**

**2ami**

**34, rue de la Tour - 84120 PERTUIS**

**✆ 90.09.65.66**

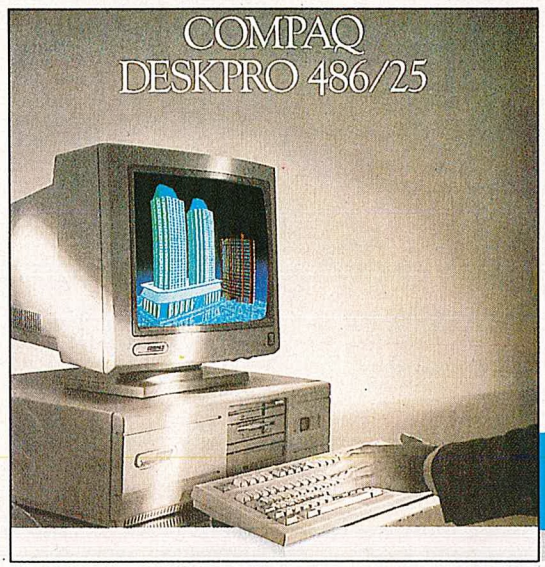
RTLink est une marque déposée POCKET SOFT Inc.

Les autres marques et autre produits sont des marques déposées de leurs propriétaires respectifs.

**Compatible : Microsoft C 6.0, Microsoft Basic 7.0, Microsoft Pascal, Microsoft Fortran, Quick C 2.6 Turbo C 1.0 & 2.0, Clarion, Quick Silver 1.2 B & 1.2 C, Clipper S 87 & V 5.0, Lattice C 34 ...**

La Dédicace
INFORMATIQUE

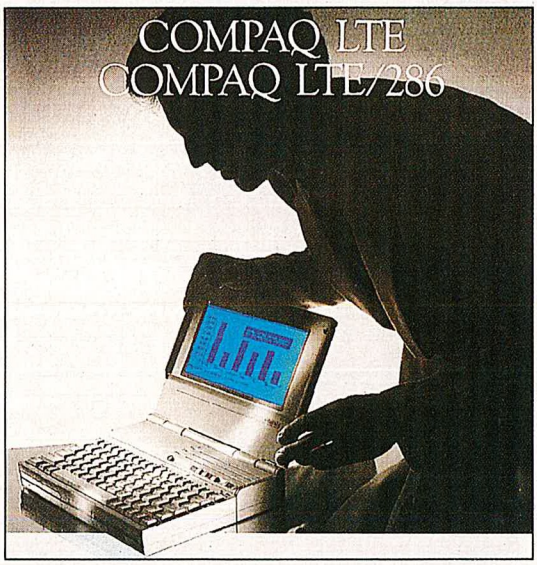
COMPAQ
Revendeur Agréé




486/25

PRÊT ET
DÉMONSTRATION
DE TOUTE LA GAMME
AUX GRANDS COMPTES

**Tél. 45.66.94.69**




LTE/286

SYSTEMPRO

**DISPONIBILITÉ**
**ET**
**LOCATION**
**IMMÉDIATES**

La Dédicace
INFORMATIQUE

173, rue de Vaugirard - 75015 PARIS
Tél. 45.66.94.69
Fax 43.06.35.18 - Télex 202 993

```
;┌──────────────────────────────────────────────────────────┐
;| HVD.ASM     (Pour programme "HVD.COM")                   |
;| ---------                                                |
;| Conversion en décimal et binaire d'un hexa introduit au clavier |
;└──────────────────────────────────────────────────────────┘

TITLE HVD

;------------------------------ MACROS ------------------------------------

NOTECURSEUR   macro                     ;;-------------------------
              mov     BH,mempag         ;; Pour la page courante,
              mov     AH,3              ;; Note position curseur
              int     10h               ;; Ligne]->DH, Colonne]->DL.
              mov     memcur,DX         ;; Mémorise ligne et colonne.
              endm                      ;;-------------------------

                                        ;;                 ┌------┐
METCURSEUR    macro                     ;;-----------------|DH/DL|
              mov     BH,mempag         ;; Dans la page courante.└------┘
              mov     AH,2              ;; Place le curseur en
              int     10h               ;; Ligne:DH, Colonne:DL.
              endm                      ;;-------------------------

CR            macro                     ;;-------------------------
              mov     AX,0E0Dh          ;; Procédure
              int     10h               ;; de mise
              mov     AX,0E0Ah          ;; à la
              int     10h               ;; ligne.
              endm                      ;;-------------------------


CLEAR         macro   CoinHG,CoinBD     ;;-------------------------
              push    BX                ;; Macro d'effacement paramé-
              mov     BH,mempag         ;; trable (coins H/G, B/D).
              mov     BH,att            ;; Maintient attribut courant.
              mov     CX,CoinHG         ;; Coin haut gauche dans CX.
              mov     DX,CoinBD         ;; Coin bas droit dans DX.
              mov     AX,0600h          ;; Fonction init/écran.
              int     10h               ;; Fait un CLS précis.
              mov     BH,mempag         ;; Rappel de la page courante.
              mov     DX,CoinHG         ;; Coin haut gauche.
              metcurseur                ;; Fonction curseur.
              pop     BX                ;;
              endm                      ;;-------------------------

                                        ;;              ┌--┐
EMET          macro                     ;;-------------|DL|
              mov     AH,2              ;; Emission du caractère └--┘
              int     21h               ;; (dans DL) par le Dos
              endm                      ;;-------------------------

                                        ;;                ┌----------┐
AFFICHE       macro                     ;;---------------| Chaîne: DX |
              push    AX                ;; Empile AX.     └----------┘
              mov     AH,09h            ;; Ecriture chaîne (fin'$').
              int     21h               ;; Interruption Dos.
              pop     AX                ;; Dépile AX.
              endm                      ;;-------------------------

                                        ;;           ┌---------------┐
ECRIT         macro                     ;;-----------| Car:AL / Att:BL |
              push    DX                ;; Préserve  └---------------┘
              push    CX                ;; les trois registres
              push    AX                ;; utilisés ici.
              mov     CX,1              ;; Ecrire 1 aractère.
              mov     BH,mempag         ;; Il faut fournir No page.
              mov     AH,09h            ;; Ecriture car à la
              int     10h               ;; position curseur.
                                        ;;----------------------
              mov     AH,3              ;; Lecture position curseur.
              int     10h               ;; par interruption vidéo.
              inc     DL                ;; Colonne suivante.
              metcurseur                ;; Place curseur à la
              pop     AX                ;; Dépile les
              pop     CX                ;; trois registres
              pop     DX                ;; sauvegardés.
              endm                      ;;-------------------------

                                        ;; ┌-----------------------┐
MUL10h        macro   facteur           ;;-| Multiplie facteur par 10h |
              mov     AX,facteur        ;; └-----------------------┘
              shl     AX,1              ;; AX contient facteur x 2.
              shl     AX,1              ;; AX contient facteur x 4.
              shl     AX,1              ;; AX contient facteur x 8.
              shl     AX,1              ;; AX contient facteur x 16.
              endm                      ;;------(plus rapide que MUL)---

;------------------------------ DONNEES -----------------------------------

HVD           SEGMENT
              ASSUME  CS:HVD,DS:HVD,ES:HVD
              org     100h

START:        jmp     debut1            ; Saut au programme lui-même.

scan          db      2,3,4,5,6,7,8,9,10,11,16,48,46,32,18,33,-1
              ;       / é " ' ( - è _ ç à a b c d e f
chiffre       db      49,50,51,52,53,54,55,56
              db      57,48,65,66,67,68,69,70,-1


avis db '┌-------------------------------------------------┐',13,10
db '|      Conversion: Hexadécimal ]-> Décimal        |',13,10
db '├-------------------------------------------------┤',13,10
db '|Entrer un héxadécimal, sans shift (clavier alpha)|',13,10
db '|     Noir-> Touche tapée, Cyan-> Digit entré     |',13,10
db '├-------------------------------------------------┘'
db '  ┌---------------------┐',13,10
db '|          (C) Bernard C.Lambey - 1990          |'
db '  |  Un hexadécimal SVP  |',13,10
db '└-------------------------------------------------┘'
db '  └---------------------┘',13,10
db '          (Fin d',39,'introduction par RETURN)',13,10
db '┌---------┬-----------┬---------┬-----------------┐',13,10
db '| Entrée  | Mémorisé  | Décimal |     Binaire     |',13,10
db '└---------┴-----------┴---------┴-----------------┘',13,10,'$'

err   db      13,10
db      '┌-----------------------------------┐',13,10
db      'Un des digits n',39,'est pas hexadécimal,',13,10
db      'Sortie possible (touche Escape) ou,',13,10
db      'réentrer le nombre s',39,'il vous plait...',13,10,'$'


cadreH  db      '┌-------------------┐','$'
cadreM  db      '|                   |','$'
cadreB  db      '└-------------------┘','$'
ouinon  db        'Encore un?...(O/N)','$'
entrer  db      '|  Un hexadécimal SVP  |','$'

att     db      0
num     db      0,0,0,0,0,0,0,0
flag    db      0
mempag  db      0
memcur  dw      0
car     dw      0
mem     dw      0

;------------------------------ PROCEDURES --------------------------------

LEBON         proc    near              ;--Scan-code dans AL au départ-
              push    BX                ; Sauve BX.
              xor     BX,BX             ; et vide le registre.
compare:      mov     DI,offset scan    ; Adresse chaîne ds DI (ES=CS)
              mov     CX,11h            ; Pour seize caractères.
              repne   scasb             ; Scanne jusque idem (place>CX)
              jne     exit              ; Si pas trouvé, saute.
                                        ; ------------------------
quitte:       mov     DX,memcur         ; Replace curseur en
              metcurseur                ; position normale.
              mov     AH,04Ch           ; Fin de programme
              int     21h               ; dite 'propre'.
                                        ; ---------┌---------------┐
oui:          clear   0034h,0C4Eh       ; sur zône.|Efface fenestron|
              mov     DX,0535h          ; Ligne 5  |et place curseur|
              metcurseur                ; Col  53  └---------------┘
              mov     DX,offset cadreH  ; Adresse haut du cadre.
              affiche                   ;------------------------
              mov     DX,0635h          ; Ligne 6 ┐
              metcurseur                ; Col  53 ┘ Fenestron.
              mov     DX,offset entrer  ; Adresse du message entrée.
              affiche                   ;------------------------
              mov     DX,0735h          ; Ligne 7 ┐
              metcurseur                ; Col  53 ┘ Fenestron.
              mov     DX,offset cadreB  ; Adresse bas du cadre.
              affiche                   ;------------------------
              mov     DX,memcur         ; Replace curseur
              metcurseur                ; en position programme.
              jmp     debut2            ; On reprend le PG.
              ret                       ;
anouveau:     endp                      ;--------------------------

                                        ;      ┌-------------------┐
                                        ;------|HEX dans CX, chaîne-> DX|
METDEC        proc    near              ;      └-------------------┘
              mov     AX,'0'-1          ; Code ascii de "0" -1 (départ)
carry:        inc     AX                ; Majore l'ascii de 1.
              add     CX,BX             ; BX est fourni par CONVERTIT.
              jc      carry             ; Saut si carry=1.
              sub     CX,BX             ; Soustrait BX de CX (retenue).
              mov     BX,DX             ; Pour la suivante DX pas admis.
              mov     [BX],AX           ; Résultat-> dans la chaîne.
              inc     DX                ; Positionne chaîne car suivant.
              ret                       ;
metdec        endp                      ;--------------------------
```

# A ce prix-là ça va faire des jaloux !!!

## AS 80386SX-20 MHz

1 Mo de RAM Ext. 8 Mo — Bios AMI — 6 slots Libre — 2 ports séries, 1 port // — Support copro. 80387SX — Alimentation 200 watts — 1 lecteur 3''1/2 1.44 Mo — Boîtier Baby ou Desk — Clavier 102 touches — MS-DOS 4.01 — **Garantie 1 an sur site**

| Écran \ Disque Dur | 40 Mo AT Bus | 85 Mo AT Bus | 125 Mo AT Bus | 210 Mo AT Bus |
|---|---|---|---|---|
| Hercule | 7562 F | 8812 F | 9375 F | 11750 F |
| VGA Mono | 8000 F | 9250 F | 9812 F | 12187 F |
| VGA Couleur 800×600 | 9312 F | 10562 F | 11125 F | 13500 F |
| VGA Couleur 1024×768 | 9625 F | 10875 F | 11437 F | 13812 F |

## AS 80386-25 MHz

2 Mo de RAM Ext. 8 Mo — Bios AMI — 6 slots Libre — 2 ports séries, 1 port // — Support copro. 80387 — Alimentation 200/230 watts — 1 lecteur 3''1/2 1.44 Mo — Boîtier Baby ou Desk — Clavier 102 touches — MS-DOS 4.01 — **Garantie 1 an sur site**

| Écran \ Disque Dur | 40 Mo AT Bus | 85 Mo AT Bus | 125 Mo AT Bus | 210 Mo AT Bus |
|---|---|---|---|---|
| Hercule | 9687 F | 10937 F | 11500 F | 13875 F |
| VGA Mono | 10125 F | 11375 F | 11937 F | 14312 F |
| VGA Couleur 800×600 | 11437 F | 12687 F | 13250 F | 15625 F |
| VGA Couleur 1024×768 | 11750 F | 13000 F | 13562 F | 15937 F |

## AS 80386-33 MHz

2 Mo de RAM Ext. 8 Mo — Bios AWARD — 64 Ko cache — 6 slots Libre — 2 ports séries, 1 port // — Support copro. 80387 — Alimentation 200/230 watts — 1 lecteur 3''1/2 1.44 Mo — 1 lecteur 5''1/4 1.2 Mo — Boîtier Baby ou Desk — Clavier 102 touches — MS-DOS 4.01 — **Garantie 1 an sur site**

| Écran \ Disque Dur | 40 Mo AT Bus | 85 Mo AT Bus | 125 Mo AT Bus | 210 Mo AT Bus |
|---|---|---|---|---|
| Hercule | 11750 F | 13000 F | 13562 F | 15937 F |
| VGA Mono | 12187 F | 13437 F | 14000 F | 16375 F |
| VGA Couleur 800×600 | 13500 F | 14750 F | 15312 F | 17687 F |
| VGA Couleur 1024×768 | 13812 F | 15062 F | 15562 F | 18000 F |

## AS 80486-25 MHz

4 Mo de RAM Ext. 16 Mo — Bios AWARD — 6 slots Libre — 2 ports séries, 1 port // — Support copro. 80487 — Alimentation 200/230 watts — 1 lecteur 3''1/2 1.44 Mo — 1 lecteur 5''1/4 1.2 Mo — Boîtier Baby ou Desk — Clavier 102 touches — MS-DOS 4.01 — **Garantie 1 an sur site**

| Écran \ Disque Dur | 40 Mo AT Bus | 85 Mo AT Bus | 125 Mo AT Bus | 210 Mo AT Bus |
|---|---|---|---|---|
| Hercule | 19125 F | 20375 F | 20937 F | 23312 F |
| VGA Mono | 19562 F | 20812 F | 21375 F | 23750 F |
| VGA Couleur 800×600 | 20875 F | 22125 F | 22687 F | 25062 F |
| VGA Couleur 1024×768 | 21187 F | 22437 F | 22937 F | 25375 F |

## MÉMOIRES

**pour TOSHIBA**

| | |
|---|---|
| 2 Mo T1200XE | 2200 F |
| 2 Mo T3200SX | 2200 F |
| 2 Mo T5200 | 2000 F |

**pour COMPAQ**

| | |
|---|---|
| 2 Mo 386N | 2700 F |
| 2 Mo 386s/20 | 1800 F |

**pour Hewlett Packard**

| | |
|---|---|
| 1 Mo HPIII, IIP | 850 F |
| 2 Mo HPIII, IIP | 1400 F |
| 4 Mo HPIII, IIP | 2300 F |

## ONDULEURS

**SHANDY Sinusoïdaux Sortie RS 232**

| | |
|---|---|
| 500 VA | 5850 F |
| 750 VA | 8900 F |
| 1000 VA | 11000 F |
| 1250 VA | 12300 F |
| 2000 VA | 19900 F |
| 2500 VA | 23900 F |

## IMPRIMANTES

### STAR LC 24-15

Prix Public : 5480 F — NOTRE PRIX : 3600 F

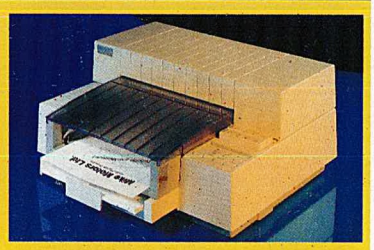
24 aiguilles — 136 colonnes — 200 CPS — 14 jeux de 14 caractères chacun — Styles d'écritures : Courrier, Prestige, Orator, Script — **Garantie 1 an retour atelier**

### STAR LASER LP4

Prix Public : 11980 F — NOTRE PRIX : 7700 F

Version Postscript : Prix Public : 16980 F — NOTRE PRIX : 10900 F

4 pages minute — 1 Mo RAM — Émulation : HP Laserjet II P — Epson FX850 — **Garantie 1 an sur site**



### HEWLETT PACKARD DESKJET 500

Prix Public : 5990 F — NOTRE PRIX : 4300 F

Impression jet d'encre — Mémoire tampon — 16 Ko - **Garantie 3 ans et 60000 pages sans panne**

ACSE

5, rue des Lyonnais - 75005 PARIS
**Tél. 16 (1) 43 37 05 06**
**Fax 16 (1) 43 37 15 46**

Frais de port non inclus.
Tous nos prix s'entendent hors taxes, T.V.A. 18,60 % en sus.

SETRI

11, bd du Gal.-de-Gaulle - 92700 COLOMBES
**Tél. (1) 47 81 42 56 +**
**Fax (1) 42 42 96 42**

```
CONVERTIT       proc    near            ;-------------------+----------
                mov     BX,-10000       ; Convertit         | Utilise
                call    metdec          ;     étape         | la
                mov     BX,-1000        ;     par           | procédure
                call    metdec          ;     étape         | "METDEC"
                mov     BX,-100         ;     le nombre     | qui reprend
                call    metdec          ;     héxadécimal   | les digits
                mov     BX,-10          ;     entré.        | hexadécim.
                call    metdec          ; Donne et          | dans
                mov     BX,-1           ;     affiche       | le
                call    metdec          ;     5 digits      | registre
                ret                     ;     décimaux.     | CX.
     convertit endp                     ;-------------------+----------

;-------------------------- DEBUT DU PROGRAMME --------------------------

DEBUT1:         mov     AH,0Fh          ; Lecture de la
                int     10h             ; page active et
                mov     mempag,BH       ; mémorisation.
                mov     AH,08h          ; Fonction lecture attribut.
                int     10h             ; Mise attribut-> dans AH.
                mov     att,AH          ; Sauve attribut courant.
                clear   0,184Fh         ;-------------------------
                mov     DX,offset avis  ; Adresse message d'entrée
                affiche                 ; pour affichage.
                notecurseur             ; Relève position curseur après
                                        ;-------------------------
     debut2:    mov     AX,0            ; Remise
                mov     BX,0            ; à zéro
                mov     CX,0            ; pour les
                mov     mem,0           ; reprises
                                        ;----------------------
                mov     CX,4            ; Entrer 4 scan-codes. ,
     ditnum:    mov     AH,0            ; Lecture au clavier.  | Lec-
                int     16h             ; Scan-code->AH,car->AL. | ture
                mov     SI,offset chiffre ; Adresse chaîne substituts.
                mov     BX,10h          ; Longueur maxi chaîne.
                sub     BX,CX           ; Calcule place où chercher.
                add     SI,BX           ; Additionne place à offset.
                mov     AL,[SI]         ; Met le bon Ascii dans AL.
     exit:      pop     BX              ; chiffre est dans AL.
                ret                     ; Retour à la boucle.
     lebon      endp                    ;-------------------------

                                        ;
DEDUIT          proc    near            ;----------|Ascii de la chaîne|
                push    DI              ;          |__________________|
                cmp     AL,30h          ; Au dessous de "0",
                jb      recom           ;       et       | Eliminés.
                cmp     AL,66h          ; Au dessus de "f",
                ja      recom           ;-------------------------
                cmp     AL,61h          ; Pour minuscules,
                jae     minus           ;       et       | correction.
                cmp     AL,41h          ; Pour majuscules,
                jae     majus           ;-------------------------
                cmp     AL,3Ah          ; Elimine ponctuation (3A-41).
                jae     recom           ;-------------------------
                sub     AL,30h          ; Tranforme ascii-chiffres
                jmp     sorded          ; en digits hexadécimaux.
                                        ;-------------------------
     recom:     mov     DX,offset err   ; On retourne au début dans les
                affiche                 ; cas éliminés, avec message.
                notecurseur             ; Lecture pos. curseur.
                clear   0034h,0C4Eh     ; Efface zône fenestron.
                mov     DX,memcur       ; Repasse zône programme,
                metcurseur              ; pour affichage message
                jmp     debut2          ; d'erreur et reprise début.
                                        ;-------------------------
     majus:     cmp     AL,47h          ; Elimine les majuscules au
                jae     recom           ; dessus de "F" et transfor-
                sub     AL,37h          ; me les autres en digits
                jmp     sorded          ; hexadécimaux et sort.
                                        ;-------------------------
     minus:     sub     AL,57h          ; Transforme Minuscule-> digit.
     sorded:    pop     DI              ; et récupère registre empilé.
                ret                     ; (le digit est dans AL).
     deduit     endp                    ;-------------------------

                                        ;
ANOUVEAU        proc    near            ;----------|Avec Oui ou Non|
                notecurseur             ; Pos.curseur.|______________|
                clear   0034h,0C4Eh     ; Efface zône fenestron.
                mov     DX,0535h        ; Ligne 5, colonne 53.
                metcurseur              ; Met curseur dans fenestron.
                mov     DX,offset cadreH ; Adresse haut du cadre.
                affiche                 ;----------------------
                mov     DX,0635h        ; Ligne 6, colonne 53.
                metcurseur              ; Met curseur dans fenestron.
                mov     DX,offset cadreM ; Adresse milieu cadre.
                affiche                 ;----------------------
                mov     DX,0735h        ; Ligne 7, colonne 53.
                metcurseur              ; Met curseur dans fenestron.
                mov     DX,offset cadreB ; Adresse bas du cadre.
                affiche                 ;----------------------
                mov     DX,0638h        ; Ligne 6, colonne 56.
                metcurseur              ; Met curseur dans fenestron.
                mov     DX,offset ouinon ; Adresse du message "O/N".
                affiche                 ;-----------------------
     entre:     mov     AH,0            ; Saisie caractère sans echo.
                int     16h             ; Interruption clavier.
                cmp     AH,01h          ; Est-ce scan de "Escape"?
                je      quitte          ; Même chose.
                cmp     AH,18h          ; Est-ce scan de "O"?
                je      oui             ; Si oui: sort. AL = caractère.
                cmp     AH,31h          ; Est-ce scan de "N"?
                je      quitte          ; Si oui: sort. AL = caractère.
                mov     DL,07           ; Sinon charge "cloche"dans DL,
                emet                    ; Emission par le Dos.
                jmp     entre           ; Et on boucle jusqu'à O ou N.
                                        ;-------------------------
     quitte:    mov     DX,memcur       ; Replace curseur en
                metcurseur              ; position normale.
                mov     AH,04Ch         ; Fin de programme
                int     21h             ; dite "propre".
                                        ; ---------+----------------+
     oui:       clear   0034h,0C4Eh     ; sur zône.|Efface fenestron|
                mov     DX,0535h        ; Ligne 5  |et place curseur|
                metcurseur              ; Col  53  +----------------+
                mov     DX,offset cadreH ; Adresse haut du cadre.
                affiche                 ;----------------------
                mov     DX,0635h        ; Ligne 6 ,
                metcurseur              ; Col  53 J Fenestron.
                mov     DX,offset entrer ; Adresse du message entrée.
                affiche                 ;----------------------
                mov     DX,0735h        ; Ligne 7 ,
                metcurseur              ; Col  53 J Fenestron.
                mov     DX,offset cadreB ; Adresse bas du cadre.
                affiche                 ;----------------------
                mov     DX,memcur       ; Replace curseur
                metcurseur              ; en position programme.
                jmp     debut2          ; On reprend le PG.
                ret                     ;
     anouveau:  endp                    ;-------------------------

                                        ;
                                        ;----|HEX dans CX, chaîne-> DX|
METDEC          proc    near            ;    +----------------------+
                mov     AX,'0'-1        ; Code ascii de "0" -1 (départ)
     carry:     inc     AX              ; Majore l'ascii de 1.
                add     CX,BX           ; BX est fourni par CONVERTIT.
                jc      carry           ; Saut si carry=1.
                sub     CX,BX           ; Soustrait BX de CX (retenue).
                mov     BX,DX           ; Pour la suivante DX pas admis.
                mov     [BX],AX         ; Résultat-> dans la chaîne.
                inc     DX              ; Positionne chaîne car suivant.
                ret                     ;
     metdec     endp                    ;-------------------------

CONVERTIT       proc    near            ;-------------------+----------
                mov     BX,-10000       ; Convertit         | Utilise
                call    metdec          ;     étape         | la
                mov     BX,-1000        ;     par           | procédure
                call    metdec          ;     étape         | "METDEC"
                mov     BX,-100         ;     le nombre     | qui reprend
                call    metdec          ;     héxadécimal   | les digits
                mov     BX,-10          ;     entré.        | hexadécim.
                call    metdec          ; Donne et          | dans
                mov     BX,-1           ;     affiche       | le
                call    metdec          ;     5 digits      | registre
                ret                     ;     décimaux.     | CX.
     convertit endp                     ;-------------------+----------

;-------------------------- DEBUT DU PROGRAMME --------------------------

DEBUT1:         mov     AH,0Fh          ; Lecture de la
                int     10h             ; page active et
                mov     mempag,BH       ; mémorisation.
                mov     AH,08h          ; Fonction lecture attribut.
                int     10h             ; Mise attribut-> dans AH.
                mov     att,AH          ; Sauve attribut courant.
                clear   0,184Fh         ;-------------------------
                mov     DX,offset avis  ; Adresse message d'entrée
                affiche                 ; pour affichage.
                notecurseur             ; Relève position curseur après
                                        ;-------------------------
     debut2:    mov     AX,0            ; Remise
                mov     BX,0            ; à zéro
                mov     CX,0            ; pour les
                mov     mem,0           ; reprises
                                        ;----------------------
                mov     CX,4            ; Entrer 4 scan-codes. ,
     ditnum:    mov     AH,0            ; Lecture au clavier.  | Lec-
                int     16h             ; Scan-code->AH,car->AL. | ture
```

# UN "POIDS MOUCHE" À MÉMOIRE D'ÉLÉPHANT !

# Note Book

NOTEBOOK 2100/286
**13 950 F**

NOTEBOOK 2200/386 SX
**16 950 F**

## NOTEBOOK 2100 - 2200 :

### *L'Ordinateur portable autonome*

Une puissance de calcul disponible avec processeur 80286 ou 80386 SX. Sa mémoire extensible de 1 à 4 Moctets supporte tous les logiciels du marché et notamment le nouvel environnement WINDOW 3. Son écran éclairé par l'arrière donne de très hauts constrastes à la résolution VGA et grâce à ses 2 sorties séries, la communication vers le monde extérieur est très simple. Minitel, échange de fichier... Avec son disque dur de 20 Mo (option 40 Mo) à accès rapide, vous stockez tous vos programmes et fichiers.

## NOTEBOOK 2100/286 : 13 950,00 F HT

Processeur 80C286 – 12 MHz. Disquette 3,5" 1,44 Mo, disque 20 Mo (25 ms). Mémoire 1 Mo (ext. 4 Mo). Ecran VGA LCD CCFT Paperwhite 32 niveaux de gris 640 x 480. Clavier AZERTY Français 85 touches. Sorties : 2 séries, 1 imprimante, Moniteur VGA externe, lecteur 5"25 et clavier numérique externe. Alimentation batterie, autonomie 2 h 30. Livré avec sacoche, bloc externe 110 V/220 V. MS DOS 4.01.

## NOTEBOOK 2200/386 SX : 16 950,00 F HT

Processeur 80386 SX - 20 MHz (caractéristiques identiques au modèle 2100).

LES NOTEBOOKS 2100/2200 sont fabriqués par NTC à TAIWAN et bénéficient de la plus haute technologie de ce fabricant. Avec une capacité de production de 8000 systèmes par mois, NTC se situe parmis les tout premiers fabricants mondiaux d'ordinateurs portables.

## SUPERGARANTIES ADITEM

Maintenance gratuite sur site dans un délai de 12 heures (pièces et main d'œuvre), sur simple demande dans toute la France. Satisfait ou remboursé pendant 10 jours. Essayez donc nos NOTEBOOKS !

## Points de vente et démonstration

**ADITEM - Tour Pleyel**
153, bd Anatole-France
93521 ST-DENIS CEDEX
tél. (1) 49 33 73 00
fax (1) 48 09 10 09

**ADITEM Service**
78, rue E. Herriot
69002 LYON
tél. 78 37 37 63
fax 78 38 07 25

**MICROBOUTIQUE**
37, passage de l'Argue
69002 LYON
tél. 78 37 46 17
fax 78 38 07 25

**ADITEM Service**
9, rue Necker
42000 ST-ETIENNE
tél. 77 32 35 08
fax 77 38 02 71

Commandez en direct. Livraison sous 48 heures. BON DE COMMANDE à retourner à : ADITEM DIRECT - 78, rue Herriot - 69002 LYON - Tél. (16) 78 37 37 63 - Fax (16) 78 38 07 25

ADITEM
DIRECT

MEMBRE DU GROUPE ADITEM INTERNATIONAL
78, rue Herriot - 69002 LYON
78 37 37 63

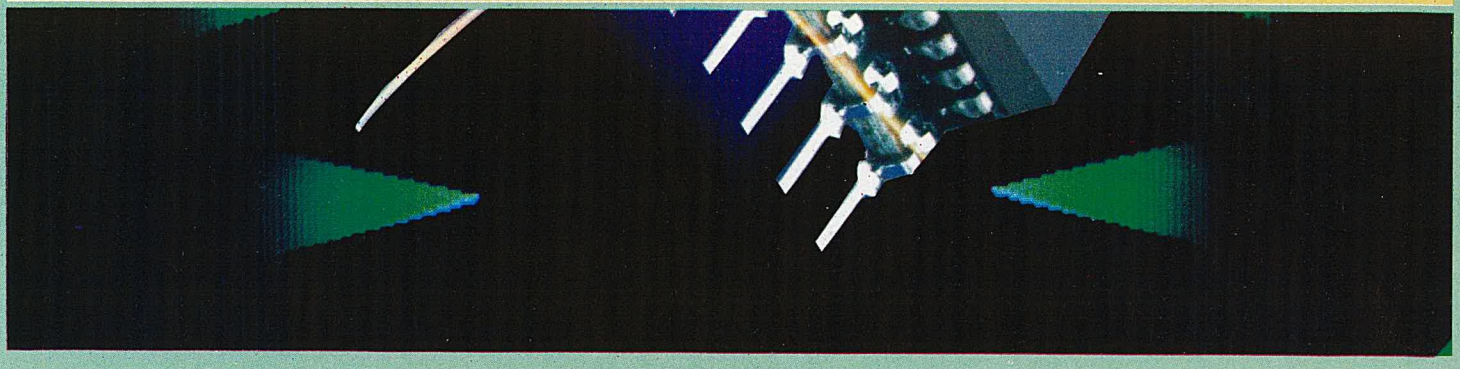
| Modèle | Qté | Prix Unit. H.T. | Prix TOTAL H.T. |
|---|---|---|---|
| NOTEBOOK 2100 ___ | ___ | 13 950 | ___ |
| NOTEBOOK 2200 ___ | ___ | 16 950 | ___ |
| | | TOTAL H.T. | ___ |
| | | Frais de livraison FORCE 12 | 280 |
| | | T.V.A. 18,6 % | ___ |
| | | TOTAL T.T.C. | ___ |

Je désire régler par :
☐ chèque bancaire
☐ demande de crédit
☐ je souhaite recevoir une documentation

DATE _____ SIGNATURE

NOM _____
Société _____
Adresse _____
Code Postal _____ Ville _____
Tél. _____

```
        cmp     AL,ØDh          ; Est-ce un CR?         | et
        jne     poursui         ; Non▶ Continuer.       | écri-
        jmp     verscr          ; Oui▶ Sortie boucle.   | tu-
poursui: cmp    AL,1Bh          ; Est-ce un ESCAPE?     | re
        je      escape          ; Oui▶ Sortie du PG.    | du
        jmp     noir            ; Saute la sortie.      | carac-
escape: jmp     quitte          ; Hors limites de "je"  | tère
noir:   push    CX              ; Sauve cpte de boucle  | conte-
        mov     BL,att          ; Sur attribut courant, | nu en
        and     BL,11110000b    ; met l'encre au noir.  | AL.
        ecrit                   ; Ecrit AL et avance.   ┘
                                ;-----------------------
        mov     AL,AH           ; Scan-code dans AL pour suite.
        call    lebon           ; Conversion du scan code.
        add     BL,ØBh          ; Encre cyan.
        inc     mem             ; Incrémente compteur.
        ecrit                   ; Ecrit AL et avance.
                                ;-----------------------
        mov     DI,offset num   ; Adresse stockage des Nb-1.
        add     DI,mem          ; Ajoute compteur.
        mov     [DI],AL         ; Envoi dans la chaîne.
        mov     [DI+1],byte ptr 24h ;Place un '$' après le car.
        pop     CX              ; Rappelle compte boucle.
        loop    ditnum          ; Boucle pour 4 caractères.
                                ;-----------------------
verscr: cmp     mem,Ø           ; Pour CR sans caractères,
        jne     nondeb          ; suite si pas Ø.
        jmp     debut2          ; sinon on recommence.
                                ;-----------------------
nondeb: notecurseur             ; Note position pour y revenir.
        mov     CX,6            ;┐
        mov     AL,4            ;|
        sub     AL,byte ptr mem ;|
        mov     DL,3            ;|Décalage variable de
        mul     DL              ;|l'affichage contrôle.
        add     CX,AX           ;|  ┌──────────────────┐
unblan: mov     DL,2Øh          ;|  |Affichage de contrôle|
        emet                    ;|  └──────────────────┘
        loop    unblan          ;┘                  |
        mov     DX,offset num+1 ; Reprend adresse num ←┘
        affiche                 ; pour affichage contrôle.
                                ;-----------------------
        mov     BX,mem          ; Prend Nb de car.
        mov     byte ptr num,BL ; Place Nb de car.
        cmp     num,4           ; Si l'entrée est de 4 digits
        je      num4            ; sauter au principal, sinon
        cmp     num,3           ; si elle est de trois, faire
        je      num3            ; le décalage des octets.
        cmp     num,2           ; Idem pour 2 digits, saut
        je      num2            ; à décalage 2 octets.
                                ;-----------------------
num1:   mov     AL, num+1       ; Si un seul digit, le
        mov     num+4,AL        ; positionne à la fin.
        mov     num+3,3Øh       ; Met un Ø en 3.
        mov     num+2,3Øh       ; Met un Ø en 2.
        mov     num+1,3Øh       ; Met un Ø en 1.
        jmp     num4            ; et passe à conversion.
                                ;-----------------------
num2:   mov     AL,num+2        ; Si 2 digits, il y
        mov     num+4,AL        ; aura de même 2 dé-
        mov     AL,num+1        ; calage et 2 mise
        mov     num+3,AL        ; en place de Ø.
        mov     num+2,3Øh       ; Met un Ø en 2.
        mov     num+1,3Øh       ; Met un Ø en 1.
        jmp     num4            ; Vers conversion.
                                ;---------------
num3:   mov     AL,num+3        ; Si 3 digits, il y
        mov     num+4,AL        ; aura de même 3 dé-
        mov     AL,num+2        ; calages et donc
        mov     num+3,AL        ; seulement 1 mise
        mov     AL,num+1        ; de Ø.
        mov     num+2,AL        ; Décalage du 1 en 2.
        mov     num+1,3Øh       ; Met un Ø en 1.
                                ;---------------
num4:   mov     AL,num+1        ; Passe 1er ascii dans AL.
        call    deduit          ; Ascii -▶ Digit.
        xor     BH,BH           ; Ø dans BH.
        mov     BL,AL           ; Pour avoir produit ALx1Ø.
        mul1Øh  BX              ; Produit. Résultat dans AX.
        mov     CH,AL           ; Sauvé dans CH.
                                ;-----------------------
        mov     AL,num+2        ; Passe 3ème ascii dans AX.
        call    deduit          ; Ascii -▶ Digit.
        add     CH,AL           ; CH muni du 2ème digit.
                                ;-----------------------
        mov     AL,num+3        ; Passe 4ème ascii dans AX.
        call    deduit          ; Ascii -▶ Digit.
        xor     BH,BH           ; Ø dans BH.
        mov     BL,AL           ; Produit ALx1Ø.
        mul1Øh  BX              ; Résultat dans AX.
        mov     CL,AL           ; 3ème digit en place.
                                ;-----------------------
        mov     AL,num+4        ; Passe l'ascii dans AX.
        call    deduit          ; Ascii -▶ Digit.
        add     CL,AL           ; Pour avoir produit ALx1Ø.
        mov     car,CX          ; ┌──────────────────────┐
                                ; |CX contient les 4 digits|
                                ; └──────────────────────┘
        mov     DX,offset num+1 ; Adresse chaîne après intro.
        call    convertit       ; Décimalise la chaîne.
        mov     DX,memcur       ;┐
        mov     DL,24           ;|Replace le curseur
        metcurseur              ;┘pour affichage décimal.
        mov     CX,5            ; 5 caractères à afficher.
        mov     BH,mempag       ; Rappel de page,┌──────────┐
        mov     BL,att          ; de l'attribut, |Aff.couleur|
        and     BL,11110000b    ; papier masqué, └──────────┘
        add     BL,14           ; et encre mise en jaune.
        mov     DI,offset num+1 ; Adresse de la chaîne pour
decimal: mov    AL,byte ptr [DI] ; affichage du décimal,
        ecrit                   ; qui est chargé dans AL,
        inc     DI              ; et on passe au suivant.
        loop    decimal         ; ┌──────────────────────┐
        mov     att,BL          ; | Conversion en binaire |
                                ; | et affichage deux bits|
        notecurseur             ; |     par deux bits.   |
        mov     DL,34           ; └──────────────────────┘
        metcurseur              ; En colonne 34.
        pushf                   ; Sauve flags courants.
        mov     DL,byte ptr car+1 ; Prend Octet lourd du nombre.
        mov     CX,16           ; Prépare 16 rotations.
        mov     BL,att          ; Prend attribut courant.
unbit:  rcl     DL,1            ; Passe bit gauche en carry.
        mov     AL,Ø            ; Vide AL.
        adc     AL,3Øh          ; Totalise Ø+48+carry= Ø ou 1.
        xor     BL,1            ; Bascule encre1-encre2,
        ecrit                   ; pour émission.
        cmp     CX,9            ; Est on au bit 9?
        jne     sq              ; Non? On saute pour boucler.
        mov     DL,byte ptr car ; Oui? Prend O-léger/Nb ┐ Après
        mov     BL,att          ; Met attribut courant, | les 8
        mov     AL,2Øh          ; et charge un espace,  | 1ers
        ecrit                   ; pour émission.        ┘ bits.
sq:     loop    unbit           ; Boucle si CX non nul.
                                ;-----------------------
        popf                    ; Dépile les flags,
        cr                      ; Alinéa pour le trait.
        mov     CX,51           ; 51 fois
trait:  mov     DL,ØC4h         ; le caractère '-',
        emet                    ; soit
        loop    trait           ; un trait.
        cr                      ; Alinéa de fin.
        call    anouveau        ; Pour reprise éventuelle.

        ends                                    HVD
        end     start
```



# Une SOLUTION INGENIEUSE à vos besoins d'utilisateur par la sélection des meilleurs sharewares!

## ☑ DP Tool Club

**5"1/4 ➠ 25F ttc**
**3"1/2 ➠ 30F ttc**
**Disquettes gratuites:**
1 pour **10**, 3 pour **20** , 5 pour **30**

- **Les toutes dernières versions**
- **Une sélection basée sur la qualité**
- **Un service à l'écoute de l'utilisateur**

Contact direct avec les auteurs, téléchargement quotidien des US, mise à jour immédiate des titres

Notre catalogue complet contient 52 pages et décrit 500 disquettes, il est envoyé contre 3 timbres. "Shareware" veut dire libre-éssai. Si après avoir essayé un logiciel diffusé en shareware vous l'utilisez régulièrement, réglez votre licence à l'auteur, elle n'est pas comprise dans nos tarifs de diffusion.

## BUREAUTIQUE & GESTION

**Fastfile+ Memo Master** : Récents programmes de gestion de fichiers, simples d'emploi. 733

**Solvit !** 4.0: Un calculateur financier programmable avec une gestion de budget intégrée. 1047

**TypeSetter PC**: programme simple de mise en page avec polices pour FX (écran CGA). 1217

**SR Info 3.4** Un SGBD relationnel, avec un compilateur, lit les fichiers dBF, Clipper, Fox. 729a,729b

**SageWords** 4.0: Excellent traitement de textes et éditeur pour programmeurs. 1024

**ProQube** Lite1.0: Tableur qui permet la manipulation de feuilles 3-D, commandes à la 123! 1718

**Simply Labels** 3: Pour gérer l'édition d'étiquettes sans soucis. Documentation en Français. 1017

**Wampum** 4.2: Gestion de fichiers à la dB3. En tête des ventes aux USA. Il faut un disque dur. 701

**Cap 90** & **Gestion de Budget**: Une compta pour particulier ou associations 1901 et une gestion de buget familial. Documentation en français. 1040

**3-B-90:** Un puissant gestionnaire de fichier au standard db3 . Simple d'emploi. Tout en français. 728

**Galaxy Lite 1.6:** Traitement de textes, rapide, puissant et simple d'emploi. 1012

## JEUX

**Win 3.0 Games:** Jeux de MahJong, d'echecs, de poker, et de backgammon pours Win 3.0 1463

**Cyrus Chess & PC-Chess:** Jeux d'echecs 2D / 3D en EGA. Superbe. 822

**Power Chess** 5.3 : Jeux d'échec avec bibliothèque d'ouvertures. Ecran graphique. 826

**Pianoman** 4.0 : Pour créez des mélodies, et les compiler avec plusieurs exemples. Doc Français. 827

**Othello's**: Jeux compilés par un spécialiste de la Fédération Française. Pour grosses têtes. 817

**Dracula in London, Kingdom of Kroz**: Deux jeux d' actions et d'aventures graphiques (CGA). 824

**Keen, Pharao's Tomb** et **Monument of Mars:** Trois jeux d'arcades EGA/VGA plutôt pour AT. 829

**Comic 4.0, Invaders et Snarfs**: Compilation des meilleurs jeux d'arcades EGA/VGA. 805

**Dungeon's of Kairn:** aventure de type donjons et dragons. Belle finition. CGA. 815

## EDUCATIFS

**Chemical** 4.0 + **Periodic**: E.A.O. de chimie, pour visualiser des molécules en 3D (EGA/VGA), et la table périodique des éléménts informatisée. 982

**Math-Tool et Simon:** Outils d'étude, de traçage de fonctions et de modélisation mathématique. 956

**The Integral Scientist**: Base de données de physique-chimie et programmes utilitaires. 981

**Mercury** 2.04: Un système complet pour résoudre les équations et faire le tracé de fonctions. 955

**The Universe**: Diaporama en couleur de belles photos d'astronomie. (EGA ou VGA). 1403

**World** 2.99 **& Geoclock** 4.2: Atlas et horloge géographiques pour mode CGA à VGA. 989

**Skyview** 2.1 : Planétarium pour AT CGA-VGA. 905

**Sky Globe** 2.0 : Un des meilleurs programmes d'aide à l'observation astronomique (CGA à VGA). 900

## PROGRAMMATION

**Qedit** 2.10: Un excellent éditeur programmable, piloté par menu avec aide en ligne. 1018

**Asic** 2.0: compilateur basic en shareware, avec son éditeur intégré. Idéal pour s'initier 102

**Modula 2** (Fitted Software) : Compilateur, éditeur et linker complets. 348,349

**A86/D86** 3.22 : Assembleur et débogueur complet avec macros. Excellent. Bien documenté. 351

**Dos Technical Reference Manual** v1.8 : 580 Ko de textes qu'il est bon d'avoir sous la main. 108

**Interrupt List** 2.91 : 430 pages d'informations sur les interruptions DOS de 00 à FF. 110

**Disassembler's ToolKit:** Utilitaires pour ceux qui s'intéressent au "reverse engineering". 308

**DeSmet C**: compilateur C fonctionnel, rapide et **Lessons in C**; un cours complet sur le C. 346

**C Tutor** 2.4: Didacticiel complet pour acquerir progressivement la maîtrise du C. 104,105

**C ↔ Tutor** 2.0 : Un cours complet pour s'initier aux finesses du C++, il faut déjà connaitre le C. 115

**Pascal tutor** 2.4: Didacticiel pour TPascal 4.0 à 5.5, approfondit la programmation objet. 512

**EZ Windows, PD menu :** fonctions QB 4.x pour la gestion de menus déroulants et fenêtres. 414

**Advanced Basic for QB 4.x**: Plus de 100 routines en ASM pour compléter le QBasic. 401

**QB4 Tree** 4.2: Séquentiel Indexé pour QB 4.x. Gère jusqu'à 10 clés sur 10 fichiers ouverts. 411

**Basic Wizard**: bibliothèques en ASM. 400+ routines pour tout faire (QB 4.x/Bascom 6). 413

**Tegl Windows Tools for** TP 5.5: pour incorporer facilement une interface graphique de pro. 532

**C-Task** 2.2: Noyau multitâches en temps réel pour programmeurs en C. 607

**C Window Boss**: Package pro. pour la gestion de fenêtres et l'interface utilisateur. 615a,615b,615c

**Super-C Lib**: Bibliothèque pour la gestion des périphériques du PC à partir du M-C et TC. 621

**TCXL** 5.5: Plus de 280 fonctions étendues pour tous les C et le C++, une référence. 614a,614b

**Code Source:** 3 jeux avec leur code; Un Tetris et GNU-chess en (C) et un Defend (asm). 111

**Snap** 4.02: Système documentaire pour programmes dB3 avec traçé d'organigramme, table de référence et vérification de structure. 717

## UTILITAIRES

**ScanVirus 76** de Mc Afee et autres anti virus, pour éliminer plus de 340 virus. Un Must. 1331

**PKZip** 1.10 + utilitaires: Le nouveau standard du compactage de fichiers. Doc. française. 1307

**Shez** 6.0, **StupenDOS**: Pour manipuler, éditer, convertir les fichiers compactés; **Lharc 2.11** qui se place parmi les premiers compacteurs. 1308

**PAK** 2.51 et **ARJ** 1.1: 2 Nouveaux compacteurs, plus puissants que ZIP, les successeurs? 1065

**Minitel ToolBox**: Emulateurs, serveurs minitel et des outils pour les gérer au mieux. 1039a,1039b

**Fast Menu** 5.0 :Gestion du disque dur par menu, et calculatrice, blocnote,agenda ... 1714a,1714b

**Win 3.0 Icons**: Une collection de 500 icones et deux éditeurs pour en créer d'autres... 1459

**Desktop Manager** 3.21 **et Navigator** 2.51 pour améliorer la gestion de fichiers par Win 3. 1460

**Easy Format** 3.2: Formate vite, en tache de fond, sur 2 lecteurs, même différents 3" ou 5". 1719

**TreeView+Automenu** 4.7: Impensable de s'en passer. Rend le DOS presque agréable. 1299

**Vmix** 2.52 : Un multitâche qui mîme Unix pour 286 et 386 (gère le mode protégé). 340

**Image Print** 4.0: Obtenez une qualité Laser sur des 9 ou 24 aiguilles, et 25 polices. 1214a,1214b

**ExtraDos Utilities** 2.30: Ajoutez 28 commandes au DOS, et simplifiez vous la vie! 1318

## GRAPHISMES

**PC Draft II** 3.26: Programme simple de D.A.O. et grapheur,en CGA haute résolution. 1407

**Graphic Workshop** 5.1: Pour visualiser, convertir, imprimer les images de format usuel. 1710

**PrintMaster/Shop** Clip Art: Bibliothèque de plus 2500 symboles & images. 1422a,1422b,1422c

**Cooper Art**: Bibliothèque de 1600 images au format .ART pour Fisrt Publisher. 1434,1435

Bibliothèque d'images de haute qualité au format PCX pour les traitements de textes ou P.A.O. les plus courants.

**Aviation** 1379 **Objets Divers** 1375 à 1378
**Maisons** 1372,1373,1374 **Femmes** 1367,1368
**Education,Ecole** 1352 **les enfants** 1356
**Voyages** 1359,1360 **Les Affaires** 1357,1358
**Hommes** 1365,1366 **Sports** 1361,1362

☞ Chaque numéro implique le paiement d'une disquette. (Par exemple 614a,614b=2x25F ou 2x30F)

Bon de commande à retourner accompagné de votre réglement à:
DP Tool Club , B.P. 745, 59657 Villeneuve d'Ascq

Nom:_____ Prénom:_____

Adresse:_____

Code.Postal:_____ Ville:_____

❑ Catalogue 52 pages (Joindre 3 timbres ) Disquette: ❑ 3"1/2 ❑ 5"1/4

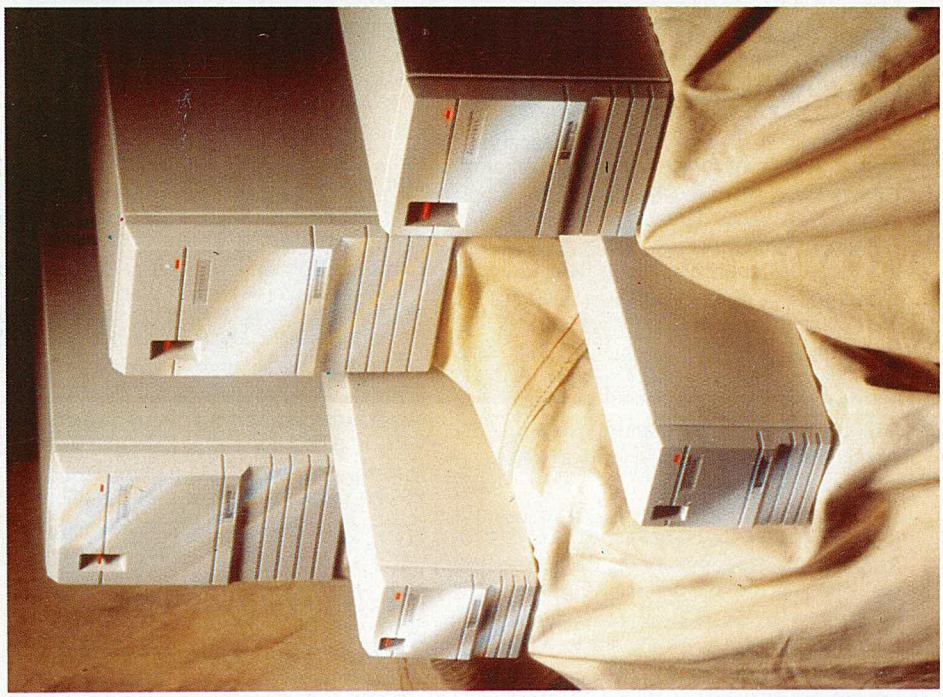
Une facture avec TVA est fournie avec chaque commande
Frais de port en sus (France, Belgique et Suisse): 16 FF ttc / envoi; commande minimum: 100 F.

# Nantucket .5

**La version française de CLIPPER 5.0 est maintenant arrivée dans tous les ateliers de développement désireux de continuer à évoluer.**

**Quand on n'avance pas, on recule, surtout dans le milieu de la Micro Informatique.**

DOMINIQUE BRUYERE



Enfin il est là, flambant neuf. Le CLIPPER nouveau est arrivé. On le tourne et on le retourne, on l'ouvre délicatement.

On risque bien de s'en servir pendant quelques années, autant en prendre soin.

Notre tête doit ressembler à celle d'une poule qui aurait reçu le dernier modèle du mini-four RIVOIRE et COMBULUZIER avec tourne-broche incorporé. Où devons-nous l'installer ?, Comment le brancher ?, Que devons-nous y faire cuire, n'y aurait-il pas un piège quelque part.

Bon... A l'eau tout le monde. Nos disquettes, (5), de sauvegarde sont prêtes. Nous copions les disquettes originales et les range dans notre coffre-fort. La disquette n° 1 dans le lecteur A:; a:install. C'est parti... Première surprise : CLIPPER 5.0 nous demande si nous avons encore 5 242 880 octets de libres dans notre poulailler. Quelle gourmandise !.

Nous engouffrons les disquettes les unes après les autres, sans nous tromper. Le message de succès final apparaît après 30 minutes. Sur un 386 à 33 MHz c'est un peu long, mais quand on aime, on ne compte pas.

Un petit tour sur le disque nous apprend que CLIPPER 5.0 s'est installé en plus de son répertoire propre, pas moins de 11 sous-répertoires. Bigre ! Il va falloir jouer avec les slashes.

Le bureau est recouvert par les documentations.

La plus grosse :

Le CLIPPER 5.0 référence, ensuite

Le guide des utilitaires et de la programmation

Le carnet de référence

Le manuel du DEBUGGER

Le manuel d'initiation

Le guide des produits et services, plus une vingtaine de pages de documentation supplémentaire à imprimer soi-même. On a de la lecture pour les longues soirées d'hiver !.

Après une rapide étude, qui va aller en s'affinant au cours de nos prochains articles, des différences fondamentales apparaissent entre CLIPPER 87 et CLIPER 5.0. Heureusement, car autrement pourquoi avoir sorti une nouvelle version.

Ces différences apportent-elles des avantages au développeur ? Toutes les promesses sont-elles tenues ? Nous allons faire le tour du propriétaire.

1) CLIPPER n'est plus seulement un compilateur. Il comporte, intégré, un préprocesseur.

2) Il est équipé, maintenant, d'un debugger qui ressemble à un vrai.

3) La compilation conditionnelle est devenue une réalité.

4) RTLINK, le nouveau linker associé, pratique la gestion dynamique des overlays, permet la fabrication de librairies runtime ou précompilées. Il travaille, bien sûr, de façon incrémentielle, comme les nouveaux linkers dont nous avons parlé dans le précédents numéros de la Revue.

5) Certaines variables reçoivent des capacités de vie et d'action, non plus pendant l'exécution du programme, mais pendant la phase de compilation.

6) Les arrays ou tableaux sont devenus multidimensionnels et dynamiques.

7) Deux nouveaux types de données voient le jour : le type NIL et le déjà fameux code block ou bloc de code.

8) Une première percée est faite en direction de la POO : (Programmation Orientée Objet).

9) L'accès à des tables d'un format différent de nos actuels DBF est désormais possible ou en passe de l'être.

10) Les fonctions définies par l'utilisateur et les procédures peuvent être déclarées STATIC et de ce fait n'être accessibles que par des appels effectués dans le même PRG.

11) Des commandes définies par l'utilisateur viennent renforcer les fonctions définies par l'utilisateur.

12) Apparition de nouveaux opérateurs puissants.

13) Norton Guide livre en série pour une documentation en ligne.

14) Les librairies ont été réécrites pour leur donner une granulomètrie plus fine.

15) Nous gardions le meilleur pour la fin : le VMM, Virtual Memory Management ou Gestionnaire de Mémoire Virtuelle.

Voilà en 15 points un résumé des modifications. Nous pouvons vous dire que ces modifications apportent toutes des améliorations.

Est-ce que ces améliorations, avec les difficultés probables d'assimilation qu'elles vont créer, étaient celles attendues par les programmes CLIPPER ? Peut-être pas à première vue, surtout si le premier regard tombe sur un extrait de code comportant la surprenante syntaxe qui entoure les codes blocks.

En fait NANTUCKET est allé beaucoup plus loin que ce que les utilisateurs espéraient, d'où un léger désapointement passager. En fouillant un peu dans les nouvelles fonctions et dans les exemples fournis par NANTUCKET, on découvre des petits trésors.

Des fonctions, prêtes à l'utilisation sans modifications majeures, sont là pour vous aider à bâtir une application, gérer les couleurs, les menus déroulants, les tableaux multidimensionnés, etc... Un peu de temps passé à décortiquer tous ces exemples vous en fera regagner très vite beaucoup.

La meilleure façon d'aborder CLIPPER 5.0 est de comparer le code d'une fonction écrite en CLIPPER 87 et traduite ensuite en CLIPPER 5.0. Voilà une bonne façon de faire connaissance avec la Librairie GRUMPFISH, écrite par un spécialiste Américain de CLIPPER, toutes versions confondues : Greg Lief. En plus de ses qualités propres, cette librairie est livrée avec ses sources dans les deux versions.

Elle se compose de métafonctions prêtes à l'emploi, bourrées d'astuces et réellement efficaces : calculette, répertoire et composeur téléphonique, calendrier et carnet de rendez-vous multi-utilisateurs, carnet de notes, chronomètre en tâche de fond, un outil simple à utiliser de génération d'écrans de saisie et d'aide en ligne, un browse très complet doté d'un module de recherche par l'exemple, query by example pour les bilingues.

Des fonctions de création et de destruction de boîtes de toutes sortes, de génération de menus ultra simples, encore une fois, à utiliser, avec des variations sur le thème de ACHOICE poussé dans ses retranchements vous rendent la vie plus que facile. La programmation réseau n'a pas été oubliée ainsi que la gestion des couleurs, à la portée de vos utilisateurs qui peuvent déplacer tout ce qui se trouve sur l'écran là ou bon leur semble et utiliser leurs couleurs préférées dans chaque module.

La gestion des chaînes de caractères à l'écran vous réserve quelques surprises, de même que la série incroyable de fonctions pour effacer un écran. Des

## *En librairie*

*— Pour ce qui est des livres en Français 2 seulement sont en notre possession à l'heure actuelle.*

*• Le premier : CLIPPER par l'exemple paru au Editions SYBEX, est une œuvre originale de Daniel Rougé. Il est relié, ce qui est important pour un livre de référence, à consulter souvent.*

*Accompagné de sa disquette qui reprend tous les exemples du livre, ce qui est une bonne façon de nous faire gagner du temps quand il s'agit de tester une fonction concoctée par l'auteur.*

*C'est une excellente introduction à CLIPPER 87 et même un peu plus. Malheureusement il porte sur sa couverture les mots "Versions 87 et 5". Là, le bât blesse. Il est évident que ce livre est sorti trop tard pour CLIPPER 87 et trop tôt pour CLIPPER 5.*

*Nous pensons que son éditeur, bien connu des informaticiens, l'a un peu trop poussé à être "le premier à sortir". Daniel Rougé n'a sûrement pas dit son dernier mot et mettre sur les présentoirs un addendum limité mais avec une disquette "bourrée" de ses connaissances ne serait peut-être pas idiot. A suivre...*

*Le deuxième livre en Français est une traduction de l'Allemand, du livre de Dirk Larish, mis sur le marché par un autre éditeur qui est loin, lui aussi d'être un inconnu : les Editions MICRO APPLICATION. Il est paru plus tard, donc il est plus complet sur CLIPPER 5. Ce qu'il faut dire... La grande différence avec le livre de Daniel Rougé est qu'il ne contient pas de code CLIPPER.*

*Ce qu'un programmeur, qui détient une version estampillée, et donc la documentation complète, attend ce n'est pas une reprise des commandes et des fonctions contenues dans cette documentation. Par contre, bravo pour les conseils de compilation et de link, ainsi que ceux évidents, donc les plus souvent oubliées par les programmeurs, sur les bonnes méthodes de programmation.*

*Le troisième en notre possession est le CLIPPER PROGRAMMING GUIDE de Rick Spence. Nous dirons que son seul défaut, pour ceux qui ne maîtrisent pas la langue de Mickaël Jackson, est justement d'être écrit dans cette langue. Pour les autres, c'est un petit Paradis. Si la librairie qui le diffuse en France et que nous avons rencontré au PC FORUM, n'avait pas oublié de nous envoyer ses coordonnées, nous aurions pu vous les donner.*

*En attendant pour le commander aux States il vous en coûtera 59.95 Dollars pour le livre et la disquette d'accompagnement, plus 20 Dollars de frais de transport : Data Based Solutions, Inc., 4010 Morena Blvd, Suite 200, San Diego, CA 92117, USA.*

fonctions destinées à économiser la durée de vie de nos écrans préférés, toujours avec une bonne pointe d'humour complètent cette extraordinaire panoplie.

La librairie GRUMPFISH est entièrement traduite en Français et adaptée aux formats Français de date, de codes postaux et de normes téléphoniques.

Donc pas de semaines qui commencent le Dimanche mais bien le Lundi comme chez nous, pas de numéros de téléphone inutilisables, etc...

Pour comprendre rapidement la philosophie de CLIPPER 5, un bon exemple vaut mieux qu'un discours fumeux. Vous trouverez dans les figures qui suivent, le code pour une fonction de chronomètre, dans les deux versions. Alors que la librairie GRUMPFISH est écrite à 99 % en CLIPPER, comme par hasard, l'exemple choisi, comporte 3 fonctions de gestion de curseur en Assembleur pour la version CLIPPER 87.

Vous constaterez qu'elles ont disparu de la version CLIPPER 5.0 parce que CLIPPER 5 sait maintenant très bien gérer le curseur (et la couleur et l'affichage !).

Avec ce simple exemple, nous allons jeter un œil sur les grands changements survenus avec CLIPPER 5.0.

La fonction a été modifiée aux seules fins d'illustrer l'article.

Pardon GREG !. Les figures 1, 2, 3 et 4 présentent le code version CLIPPER 87.

Les figures 5, 6 et 7 présentent, elles, la version CLIPPER 5.0.

Les figures 8 et 9 vous donnent le résultat après passage au préprocesseur.

La figure 10 vous donne un extrait du GRUMP.CH.

Prenez des notes !.

Ces exemples ne sont pas à recopier, mais à étudier à tête reposée. Il manque des fonctions additionnelles qui auraient pris trop de place dans notre pagination réduite.

La première version n'apporte pas de remarques particulières...

Regardez quand même, comment GREG s'y prend pour mettre une couleur à l'enverse de ce qu'elle était.

Par contre nous allons suivre la seconde version ligne par ligne.

Première remarque : nouvelle façon de signaler des commentaires. Sont maintenant permises les syntaxes suivantes : * : l'étoile, en début de ligne. // la bouble barre oblique en début de ligne ou au milieu. /* commentaire */ quand il s'agit d'incorporer un commentaire important sur plusieurs lignes et les célèbres Dupond et Dupont '&&' en fin de ligne de commande.

Pour des raisons de compatibilités futures, abandonnez les Dupond et Dupont et l'étoile '*'.

Faire débuter vos en-têtes de programmes écrits en CLIPPER 5 par /* vous permettra de savoir tout de suite sur quelle version de CLIPPER vous travaillez.

# include "grump.ch". Voilà du nouveau dans le monde CLIPPER.

Cela signifie que l'on demande au préprocesseur d'inclure dans sa séance de travail un fichier appelé dans ce cas "grump.ch". Grump parce qu'il est spécifique à la librairie GRUMPFISH et "ch" pour CLIPPER HEADER ou entête Clipper. Comme vous pouvez le voir dans la figure 10, ce fichier contient des définitions de toutes sortes.

C'est un peu l'équivalent d'une longue déclaration de variables publiques qui assignerait des noms plus faciles à mémoriser à des touches de fonction ou à tout autre cible ; par exemple : PUBLIC ENTER ENTER = 13 en CLIPPER 87 et # DEFINE K_ENTER 13 en CLIPPER 5.0. La différence fondamentale avec des variables tient dans le fait que le préprocesseur "de tout s'occupe !".

Au moment de la précompilation, c'est lui qui, automatiquement, remplace le mot ENTER par 13. Il y a substitution réelle et non plus macro-substitution. Le fichier GRUMP.CH contient même les notes de la gamme, sur 6 octaves !.

Au lieu de faire des tones (xxx,xx) au résultat incertain, il suffit de mettre une suite de notes prédéfinies dans un tableau, la durée de chaque note dans la deuxième dimension de ce tableau et grâce à une petite fonction de lui faire jouer la mélodie de votre choix.

Nous nous égarons déjà en vous parlant de tableau multidimensionné, nous y reviendrons bien entendu. En attendant nous vous faisons confiance pour trouver des idées qui exploiteront toutes les possibilités de ce INCLUDE et des fichiers .CH personnalisés.

# define ALREADY_ON apparaît sur la ligne suivante pour lui attribuer la valeur logique start_time > 0. Au cours de la phase de travail du préprocesseur, celui-ci remplacera ALREADY_ON par l'expression "start_time > 0" dans le fichier qui sera soumis à CLIPPER pour la vraie séance de compilation.

Nos premières variables de type STATIC apparaissent pour définir start_time et les coordonnées d'emplacement du chronomètre. Dans la version 87, une variable PUBLIC start_time devait être déclarée pour conserver le chrono en tâche de fond ; grâce à STATIC, la valeur de start_time reste confinée dans le cadre de popstop.prg.

Sauvegardée en sortie si le chrono est mis en tâche de fond ou remise à zéro, elle reprendra sa dernière valeur lors d'un nouvel appel au chrono. C'est tout l'avantage des variables de type STATIC.

Notez la présence non plus du signe '=' mais des deux signes ':='. C'est nouveau, ça vient de sortir, du moins chez CLIPPER 5.

Ce genre de ponctuation s'appelle un "assignment on line" ce que tout le monde aura traduit par "assignement dans la ligne". C'est beau la langue Française, non ?. On peut être fier. Il y a toute une série de ces nouveaux opérateurs dans CLIPPER 5.

La première fonction rencontrée : popstop appelle tout de suite quelques réflexions. Premièrement, elle porte le même nom que le .PRG qui la contient ; les paramètres de compilation passés à CLIPPER contiennent /n et celui-ci permet d'éviter cela. Il supprime la définition automatique d'une procédure qui porte le même nom que le programme qui la contient.

C'est quand même plus facile pour s'y retrouver dans une forêt de procédures et on gagne de la place dans la table des symboles.

Deuxièmement, les parenthèses qui la suivent contiennent les paramètres qu'elle attend. C'est une nouvelle façon de passer des paramètres, incompatibles, dans la même fonction avec une déclaration PARAMETER. Nous y reviendrons, le sujet est vaste.

Troisièmement vous remarquerez que les # include, le # define et la déclaration STATIC ne sont pas situés dans la fonction, mais au-dessus. Cela signifie que ces données si elles s'appliquent à la fonction popstop, pourraient également s'appliquer à toutes les fonctions ou procédures déclarées dans le même .PRG.

Une déclaration de variables de type LOCAL se présente ensuite. Dans CLIPPER 5 il se passe toujours quelque chose !.

Pour résumer, (nous reviendrons en détail sur ce sujet également) il existe maintenant quatre types de variables : PRIVATE, PUBLIC, STATIC et LOCAL dont la visibilité et la durée de vie sont différentes. Vaste sujet qui fera l'objet d'un article entier sinon de deux.

En descendant un peu dans le programme, vous découvrirez une ligne "DO WHILE key != 27". Greg aurait pu écrire DO WHILE key != K_ESC pour faire plus couleur locale. Les deux fonctionnent et vous pouvez constater qu'à l'époque (fin 1990), où Greg a écrit cette fonction, il n'avait pas encore débarrassé son "moi" intérieur profond des vieilles habitudes bourgeoises et décadentes qui ont marqué tant de développeurs au plus profond

TEL : (16) 40 08 05 03

# OCIS & EUROPEAN CONNECTION

FAX : (16) 40 48 75 77

IMPORT US IMPORT US IMPORT US

d'eux-mêmes durant le règne de CLIPPER 87. (Vu la révolution culturelle !).

C'était l'heure de la récré ; rangez vos goûters, on continue. D'autant plus que quelques lignes plus loin, Greg pris d'une contrition soudaine, nous gratifie d'un très beau et presque émouvant ".and. key != K_ESC".

Sautons à pieds joints en bas, ou presque, de la figure n° 6 pour découvrir notre premier bloc de code.

Il est petit et ressemble à son père avec quand même les yeux de sa mère. On ne blague pas avec un code en bloc, surtout quand on n'y comprend rien ou presque !

En fait, et ce sera plus facile pour ceux qui ont suivi, au début de la fonction, la ligne "if (gfproc != NIL)", vérifiait l'état du premier paramètre passé à la fonction, en l'occurrence "gfproc", par l'intermédiaire d'une hypothétique touche de fonction.

Il faut savoir que la fonction "setkey()" assigne, sous forme de bloc de code, une routine de programme, à une touche ; cette routine sera exécutée à chaque frappe sur cette touche quand le programme est en wait staite.

Donc, pour éviter la récursivité, en entrée de fonction, le hot key est désactivée par setkey (dernière_touche, action), dans cet exemple la valeur NIL (nul) est affecté à (action).

En sortie on lui rend la possibilité de fonctionner à nouveau, pour peu qu'elle ait déjà fonctionné. Vous nous suivez ?

Dans cet exemple setkey (dernière_touche, action) où dernière_touche est la touche sauvegardée en entrée de programme et action est ce fameux code block { | p,l,v | popstop(p,l,v) }. Le code block est chargé de passer 3 paramètres p,l, et v à la fonction popstop(). C'est aussi simple que cela.

On peut définir un block de code (nous préférons l'appellation code block) comme une fonction définie par l'utilisateur que l'on pourrait traiter comme des données. Une UDF est appelée par l'application, alors qu'un code block est évalué, comme on dit maintenant ; on peut aussi comparer un code block à une macro, la différence étant que le code block est évalué à la compilation, comme du code ordinaire et donc surveillé par le compilateur et le linker, alors qu'une macro est traitée durant l'exécution en passant par l'analyseur de macro. D'où une perte de temps, d'espace mémoire et d'efficacité tout court.

Nous reviendrons en détail sur ces fameux codes blocks, le jeu en vaut la chandelle.

Arrivés à la fin de cet article, nous n'avons pas eu le temps de découvrir ce qui est en fait la différence fondamentale entre les deux versions de NANTUCKET et ce qui va réellement faire bouger les programmeurs dans leurs habitudes : les objets GET et TBROWSE avec son acolyte TBCOLUMN.

C'est en fait toute la saisie plein écran qui se trouve modifiée et améliorée d'une façon incroyable mais également toute l'interface de fenêtrage aussi bien sur les fichiers que sur les tableaux. Il y a quelques mois, nous réclamions un DBEDIT WHILE ; aujourd'hui nous et vous, sommes servis.

On n'a pas fini de faire le tour des possibilités, limitées par notre imagination, des nouvelles possibilités de browsing. Vous n'aurez quand même pas tout d'un seul coup. A suivre dans le prochain numéro, si vous le voulez bien...

En guise de conclusion, quelques réponses à quelques questions beaucoup entendues ces temps-ci.

## Quelques questions

**— CLIPPER 5.0 est-il buggué ?**

La réponse est oui, moins tout de même que ce que beaucoup aimerait faire croire et encore moins que certaines versions de DBMS numérotées .0 à leur sortie. Certains PDG ne s'en sont pas encore remis et, à l'heure actuelle, vendraient des lacets dans les couloirs du métro de New York. Si, si...

— Mauvaise nouvelle encore pour les esprits chagrins, nous pouvons vous dire, en exclusivité, que PC TECHNOLOGIE a reçu le Mercredi 3 Avril la version 5.01 et que donc elle sera disponible avant même que vous n'ayez compris comment fonctionne CLIPPER 5. Souriez...

**— CLIPPER 5 est-il entièrement compatible avec CLIPPER 87 ?**

Pour notre part, nous ne pensons pas qu'il le soit entièrement. Il l'est au prix de quelques modifications, en particulier dans les déclarations de variables. Mais quel intérêt à compiler du code CLIPPER 87 avec CLIPPER 5 ? Ce serait comme de remplir le réservoir d'une FERRARI (pour la livraison, vous faites comme d'habitude, oui, rouge, comme d'habitude, merci !), nous disions donc d'une FERRARI avec de la bière GROTAMBOUR (pas de livraison, merci).

**— CLIPPER 5 est-il un compilateur Dbase ?**

Heureusement non. C'est un langage de programmation, un vrai langage, doté de son propre compilateur et de son propre environnement. Il conserve les fichiers .DBF, mais va s'ouvrir à d'autres formats. Ceci dit, de la même façon que vous pouvez le traduire en Français, vous pouvez lui faire exécuter des commandes venues d'ailleurs, sans trop de problèmes, par l'intermédiaire du préprocesseur.

**— Comment se former rapidement ?**

En assistant à des séminaires ou conférences de développeurs les prochains à avoir lieu sont dans l'ordre d'apparition sur le calendrier.

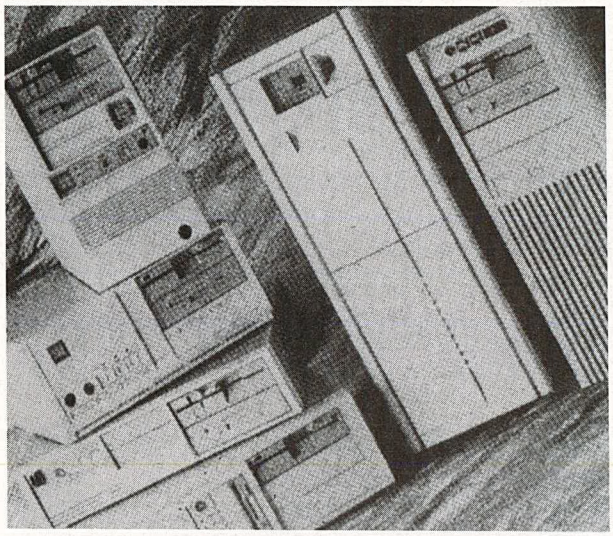
• A Paris les 4 et 5 Juin pour les débutants, les 6 et 7 Juin pour les plus "dégourdis". Organisateur : LDQS à Nice, l'un des animateurs est justement GREG LIEF, dont nous vous parlions plus haut. Renseignements : 93 16 09 12 à Nice.

• A Arles les 24, 25 et 26 Juin. Organisateur PC TECHNOLOGIE. Pas de niveaux différents, plutôt des ateliers différents. Parmi les animateurs, l'auteur de la librairie FUNCKY, nous avons nommé DIRK LESKO, un tout bon que nous aimons autant que GREG LIEF. Renseignements : 42 02 00 88 à Paris.

• Autre technique pour économiser 10 % sur le prix du séminaire de PC TECHNOLOGIE, adhérer au NUGF, le Nantucket User Group France dont nous avions donné l'adresse en Février et que certains ont trouvé difficile à lire : NUGF, 18 rue Gambetta 60000 Beauvais. Tél. : 44 45 62 19 à Beauvais.

# Que votre volonté soit faite !

## Votre micro-ordinateur sur mesure.



Solidement implantée dans le domaine de la fabrication de micro-ordinateurs depuis plus de quatre années, **CB Informatique** vous propose une configuration adaptée à vos besoins demeurant parfaitement évolutive au fil du temps.

Testés durant 72 heures dans nos laboratoires, les micro-ordinateurs CB vous assurent de la qualité professionnelle.

**Le coût ?** Nous vous en laissons seuls juges !

Voici donc quelques exemples de micro-ordinateurs CB standards pour lesquels nous nous ferons un plaisir de vous proposer un devis correspondant à vos choix.

| *Tous nos prix s'entendent H.T. départ CB.* | Hercules Monochrome | V.G.A. Monochrome | V.G.A. Couleur 800 x 600 |
|---|---|---|---|
| **CB 286 - 12** Disque dur 40 Mo AT Bus | **5 700** | **6 200** | **7 600** |
| **CB 386 SX - 20** Disque dur 80 Mo AT Bus | **11 200** | **11 650** | **13 900** |
| **CB 386 - 33** Disque dur 110 Mo AT Bus | **16 100** | **16 500** | **19 900** |

Toutes ces configurations comprennent : un boîtier 'Desktop', 1 mégaoctet de mémoire centrale, un lecteur 5,25'(1,2 Mo) ou 3,5' (1,44 Mo), les contrôleurs d'écran et de disques, un clavier 102 touches AZERTY ainsi que deux ports série et un port parallèle.

**Tarif spécial Revendeurs : nous consulter au tél : (1) 43.96.93.18 ou Fax : (1)43.96.07.44**

---

Je souhaite recevoir une proposition concernant le matériel suivant:

| Boîtier | Carte mère | Mémoire | Lecteurs | Disque dur | Écran | Souris | MS DOS 4.01 | Autres |
|---|---|---|---|---|---|---|---|---|
| *Desktop Mini-Tower Tower* | *286-12, 386 SX-20 386-20, 386-25 386-33* | *1, 2, 4, 8 Mo* | *1 Lecteur 5,25 ou 3,5 2 Lecteurs 5,25 ou 3,5* | *20,40,80,110 Mo ou supérieur* | *Hercules mono VGA mono VGA coul. 800x600* | *Oui Non* | *Oui Non* | *Par exemple : streamer, imprimante, scanner...* |
| | | | | | | | | |

Société : _____
Nom et prénom : _____
Adresse : _____
Code postal : _____
Ville : _____
Téléphone / Fax : _____ / _____

CB informatique
30, avenue du Maréchal de Lattre de Tassigny
**94410 SAINT MAURICE**
Tél : (1) 43.96.93.18
Fax : (1) 43.96.07.44

```
*:*********************************************************************
*:
*:        Programme: POPSTOP.PRG
*:
*:         Système : GRUMPFISH LIBRARY
*:         Auteur  : Greg Lief
*:       Copyright (c) 1988-89, Greg Lief
*:   Dernière modif : 19/08/89 - améliorations estéthiques et alarme
*:                    remerciements spéciaux à Dennis Levin
*:   Traduction et légères modifications dues au manque de concision de
*:   la langue française : Dominique Bruyère
*:
*:   Procs & Fncts: DRAWSTOPBX()
*:
*:   Appelle      : CURS_TOP()    (fonction Assembleur dans CURSTOP.ASM)
*:                : CURS_BOT()    (fonction Assembleur dans CURSBOT.ASM)
*:                : NEWCURS()     (fonction Assembleur dans CURSNEW.ASM)
*:                : FENETRE()      fonction de fenêtrage ajoutée pour
*:            cet article et très différente dans la librairie GRUMPFISH
*:
*:   Chronomètre pop-up
*:
*:*********************************************************************

** Pour que le chrono puisse fonctionner en tache de fond la variable
** START_TIME doit être déclarée PUBLIC AUPARAVANT

PARAM gfproc,line,var
PRIVATE oldscrn,oldcolor,elaptime,disptime,hotkey,moldrow,moldcol,ptr,;
 mcurstop,mcursbot,already_on,maincolor,mwincol,mtop,mleft,gfsavedefs

*  détermine si la fonction a été appelé via une touche de fonction
*  si c'est le cas  annulation de celle-ci pour éviter la récursivité
*  Les cinq lignes suivantes n'ont pour but que de mettre en évidence
*  l'ombre et ne font pas partie de GRUMPFISH
Setcolor("w+/r")
CLEAR SCREEN
FOR i=0 TO 24
   @ i,0 SAY Replicate("x",80)
NEXT

hotkey = (PCOUNT() > 0)
IF TYPE('stopkey')='U'   && F5 par défaut si pas déclarée autrement
   stopkey = -4
ENDIF
IF hotkey
   SET KEY stopkey TO
ENDIF

** détermine la couleur et la position de l'écran
** vérifie les variables globales
maincolor = IF(TYPE('gfstopcol') = 'U', 'W/B', gfstopcol)
mwincol   = IF(ISCOLOR(),'+W/N','I')
mtop      = IF(TYPE('gfstoptop') = 'U',   4, gfstoptop)
mleft     = IF(TYPE('gfstopleft') = 'U', 26, gfstopleft)
*                          FIGURE N°1
```

```
      DO WHILE INKEY() = 0
         elaptime = SECONDS() - start_time
         *petite musique d'alarme toutes les minutes ou toutes les heures
         IF elaptime = 60.00
            TONE(880,1)
            TONE(880,1)
            TONE(880,1)
         ENDIF
         IF elaptime = 3600.00
            TONE(440,2)
            TONE(440,2)
            start_time = 0
         ENDIF
      disptime = STR(INT(elaptime/60),2) + ':' + ;
      IF(elaptime%60<10,'0'+STR(elaptime%60,4,2),STR(elaptime%60,5.2))
      @ mtop+3, mleft+10 SAY disptime
      ENDDO
      SETCOLOR(maincolor)
      @ mtop+3, mleft+10 SAY disptime
      keypress = LASTKEY()
      IF .NOT. (keypress = 27 .AND. already_on)
         STORE .F. TO start_time, already_on
      ENDIF
   ENDIF
ENDDO
*** vérifie si des changements de configuration ont eu lieu
***.il n'y a pas eu de changements (version démo)
*IF gfsavedefs
* IF yes_no("Voulez-vous conserver les changements pendant la séance")
*      PUBLIC gfstopleft, gfstoptop, gfstopcol
*      gfstopleft = mleft
*      gfstoptop  = mtop
*      gfstopcol  = maincolor
* ENDIF
*ENDIF
** remet la hot key en route
IF hotkey
   SET KEY stopkey TO POPSTOP
ENDIF
** restaure l'environment de départ
SETCOLOR(oldcolor)
RESTSCREEN(mtop, mleft, mtop + 7, mleft + 29, oldscrn)
SET CURSOR ON
RETURN
FUNCTION drawstopbx    && dessine le boitier du chrono
SETCOLOR(maincolor)
fenetre(maincolor,mtop, mleft, mtop+6, mleft+27)
*détermine l'inverse de la couleur principale
ptr = AT('/', maincolor)
SETCOLOR(SUBSTR(maincolor, ptr+1) + '/' + SUBSTR(maincolor, 1, ptr-1))
@ mtop,    mleft+ 8 SAY 'CHRONOMETRE'
SETCOLOR(maincolor)
@ ROW(),   mleft+ 1 SAY REPLICATE(CHR(219), 7)
@ ROW(),   mleft+19 SAY REPLICATE(CHR(219), 8)
@ ROW()+3, mleft+ 9 SAY '0:00.00'
@ mtop+2, mleft+9, mtop+4, mleft+18 BOX '┌─┐│┘─└│'
RETURN('')
                          FIGURE N°3
```

```
FUNCTION fenetre
* A utiliser avec popstop.prg
* petite fonction rapide pour ombrer une fenêtre
* ne fait pas partie de la GRUMPFISH LIBRAIRIE
* Les fonctions de la GRUMPFISH sont beaucoup plus pointues.

PARAMETER couleur1,l1,c1,l2,c2
PRIVATE zone_b,zone_d,i
Setcolor(couleur1)
M->zone_b=Savescreen(l2+1,c1+2,l2+1,c2+2)
M->zone_d=Savescreen(l1+1,c2+1,l2+1,c2+2)
FOR M->i=2 TO Len(M->zone_b) STEP 2
   M->zone_b=Subs(M->zone_b,1,M->i-1)+Chr(8)+Subs(M->zone_b,M->i+1)
NEXT
FOR M->i=2 TO Len(M->zone_d) STEP 2
   M->zone_d=Subs(M->zone_d,1,M->i-1)+Chr(8)+Subs(M->zone_d,M->i+1)
NEXT
@ M->L1,M->C1,M->L2,M->C2 box "          "
Restscreen(M->L2+1,M->C1+2,M->L2+1,M->C2+2,M->Zone_b)
Restscreen(M->L1+1,M->C2+1,M->L2+1,M->C2+2,M->Zone_d)
RETURN .T.

                          FIGURE N° 4
```

```
gfsavedefs = .F.
** Coordonnées position/couleur écran modifiables par l'utilisateur.
** GFSAVEDEFS est un flag mis à .T. toutes les fois que l'utilisateur
** modifie la couleur ou la position de l'écran de la calculette
** La touche de fonction pour modifier les couleurs est ALT-F10 ,
** bien que cela puisse être modifié très facilement.Les touches de
** déplacements, déplacent justement la boite.On peut sauvegarder une
** nouvelle configuration ou conserver l'ancienne.
moldrow = ROW()
moldcol = COL()
oldscrn = SAVESCREEN(mtop, mleft, mtop + 7, mleft + 29)
oldcolor = SETCOLOR()
* Les fonctions assembleur ont été déconnectées
*mcurstop = Curs_Top()       && position du curseur en début de ligne
*mcursbot = Curs_Bot()       && position du curseur en fin de ligne
*NewCurs(32,0)               && équivalent à SET CURSOR OFF
SET CURSOR OFF
DrawStopBx()
keypress = 0
** si la variable start_time est de type numérique , c'est tout
** si  lement parce que le coucou Bavarois fonctionne.
alre .y_on = (TYPE('start_time') = 'N')
DO WHILE keypress != 27
   IF ! already_on
      @ mtop+1,  mleft+3  SAY ' Alt-F = Tâche de fond
      @ mtop+5, mleft+6  SAY '[Espace]= chrono'
      keypress = INKEY(0)
** Pour la Revue de l'Utilisateur de l'IBM-PC j'ai désactivé le
** changement de couleurs et le déplacement d'une fenêtre. Je ne
** peux quand même pas vous donner tout le code !!!

*     IF keypress = -39         && Alt-F10: change la couleur
*        maincolor = ColorPal(maincolor, 16, IF(mleft > 36, 0, 64))
*        IF LASTKEY() != 27
*           SETCOLOR(maincolor)
*           drawstopbx()
*           gfsavedefs = .T.
*        ENDIF
*        LOOP
*     ENDIF
*     IF keypress < 32
*        IF CheckMove(7, 29)   && voir _MOVING.PRG pour cette fonction
*           gfsavedefs = .T.
*        ENDIF
*        LOOP
*     ENDIF
   ENDIF
   IF keypress = 289
      start_time = SECONDS()
      keypress = 27    && pour forcer la sortie de la boucle infernale
   ELSEIF keypress != 27 .OR. already_on
      ** on utilise un nouveau message si le coucou Bavarois tourne
      @ mtop+1,  mleft+3  SAY IF(already_on, 'Chrono en tâche de fond',;
         ' Alt-F = Tâche de fond ')
      @ mtop+5, mleft+6  SAY IF(already_on,'[Espace]= chrono',;
                                '[Espace]= arrêt ')
      SETCOLOR(mwincol)
      start_time = IF(! already_on, SECONDS(), start_time)
*                          FIGURE N° 2
```

```
/*
   Programme: POPSTOP.PRG
   Système  : GRUMPFISH LIBRARY
   Auteur   : Greg Lief
   Copyright (c) 1988-90, Greg Lief
   Clipper 5.0 version
   Instruction de compilation: clipper popstop /n/w/a

   Procs & Fncts: DRAWSTOPBX()

           Appelle: SHADOWBOX()   (fonction dans SHADOWB.PRG)
                    CHECKMOVE()   (fonction dans MOVING.PRG)
                    COLORPAL()    (fonction dans COLORPAL.PRG)
   Notez au passage la façon de déclarer un commentaire.

*/

// début des directives au préprocesseur; autre façon pour un commentaire

#include "grump.ch"
#include "inkey.ch"

// fin des directives au préprocesseur
// début des déclarations globales

#define ALREADY_ON   (start_time > 0)
static start_time := 0, stoptop := 4, stopleft := 26

// fin des déclarations globales

function popstop(gfproc, line, var)
local elaptime, disptime, hotkey := 0, key := 0, tempcolor, ;
 oldscrn := savescreen(stoptop, stopleft, stoptop + 7, stopleft + 29)
GFSaveEnv()
// détermine si la fonction a été appelé via une touche de fonction
// si c'est le cas  annulation de celle-ci pour éviter la récursivité
// si le paramètre gfproc a été passé, setkey() donne à une touche une
// valeur de type bloc active.Dans ce cas une valeur nulle grâce au
// nouveau type NIL.
if (gfproc != NIL)
   setkey(hotkey := lastkey(), NIL)
endif
drawstopbx()
// si la variable start_time est de type numérique , c'est tout
// simplement parce que le coucou Bavarois fonctionne.
do while key != 27
   if ! ALREADY_ON
      @ stoptop+1,  stopleft+3  say " Alt-F = tâche de Fond "
      @ stoptop+5,  stopleft+6  say "[Espace]= chrono"
      if (key := ginkey(0)) = K_ALT_F10   // change de couleur
         tempcolor := colorpal(tempcolor, 16, if(stopleft > 36, 0, 64))
         // s'ils n'ont pas changé d'avis on change la couleur du chrono
         if lastkey() != K_ESC
            ColorSet(C_STOPWATCH_BOX, tempcolor)
            drawstopbx()
         endif
         loop
      endif
                          FIGURE N° 5
```

**LDQS 5 RUE EUGENE EMANUEL 06000 NICE TEL 93 16 09 12 FAX 93 88 08 90**

**GRUMPFISH est 100 % écrit en CLIPPER.**
DOCUMENTATION EN FRANÇAIS.

Distribué Exclusivement par L.D.Q.S
PRIX H.T. 1950 Frs + 5o Frs de port.
+1000 Frs pour les sources en Summer '87 et 5.0.
VERSION DEMO 50 frs

**TELEPATHY** : La librairie de télécommunication N°1. Ne perdez pas votre temps, il n'y a pas mieux. Avec BLINKER vous la mettez en overlay.Qui dit mieux ? Compatible Summer '87 et 5.01

Seul TELEPATHY peut :

- Etre mis en OVERLAY avec BLINKER et WARPLINK
- Vitesse jusqu'à 115200 bauds, accepte tous les protocoles
- Définition des fonctions d'attente par l'utilisateur.
- Terminal d'émulation configurable par l'utilisateur.
- Supporte plus de 2 ports de communication
- Supporte 16550 UART, DTR/DSR HANDSHAKING CTS/RTS HANDSHAKING
- Cherche un caractère dans le buffer, désinstalle l'*interrupt handler* si le programme se plante.
- Compatible NORTON GUIDES ET TOM RETTIG, et beaucoup d'autres fonctions .

PRIX H.T. 2250 Frs + 5o Frs de port.

**TOUT SUR LA TELECOMMUNICATION AVEC CLIPPER DE NANTUCKET**
**SEMAINAIRE A PARIS - NOUS CONTACTER DES AUJOURD'HUI**

Nous ne vendons pas de pipeaux (flûte à bec imitant le gazouillement des oiseaux) mais le compilateur **CLIPPER 5.01** de NANTUCKET.

Nous vous expédions uniquement la seule version utilisable par les professionnels : la 5.01. Avec bien sur inclus **l'excellent manuel en Français** : *Le grand livre de NANTUCKET CLIPPER 5 et NANTUCKET TOOLS 2.*

| | |
|---|---|
| CLIPPER 5.01 | PRIX H.T 4700 Frs + port 100 Frs |
| FUNCKY | PRIX H.T 1850 Frs + port 50 Frs |

La percée technologique continue avec BLINKER 1.5

En plus de toutes les fonctions reconnues par tous les linkers, BLINKER 1.5 vous apporte des avantages uniques :

BLINKER 1.5 vous permet indifféremment de travailler avec CLIPPER '87 ou CLIPPER 5.0

Toujours le LINKER le plus rapide,et de loin (*10 fois plus vite et même plus avec l'incrémental linking*).

**COMPRESSION DE LA TABLE DES SYMBOLES.** Tous les symboles en doubles sont effacés de votre .EXE.

Attention : BLINKER est vendu en exclusivité par L.D.Q.S. Seuls nos clients recevront les mises à jour.

PRIX H.T. 2850 Frs + 5o Frs de port.
Livraison 24 Heures

**LDQS 5 RUE EUGENE EMANUEL 06000 NICE TEL 93 16 09 12 FAX 93 88 08 90**

## BON DE COMMANDE

Nom : ..... Société : .....
Adresse : ..... Code Postal : ..... Ville : .....
Tel : ..... Fax : .....
Produit(s) : ..... Ci-joint un chèque de : ..... Frs

```
      if key < 32 .and. key != K_ESC .and. key != K_ENTER
         checkmove(key, 5, 29, @stoptop, @stopleft, @oldscrn)
         loop
      endif
   endif
   if key = K_ALT_F
      start_time := seconds()
      key := K_ESC      // pour forcer la sortie de la boucle infernale
   elseif key != K_ESC .or. ALREADY_ON
     // on utilise un nouveau message si le coucou Bavarois tourne
     @ stoptop+1,stopleft+3  say if(ALREADY_ON, "Chrono en tâche de fond",;
        " ALT-F = tâche de Fond ")
      @ stoptop+5,stopleft+6 say if(ALREADY_ON, "[Espace]= chrono",;
        "[Espace]= arrêt ")
      ColorSet(C_STOPWATCH_WINDOW)
      start_time := if(! ALREADY_ON, seconds(), start_time)
      do while inkey() = 0
         elaptime := seconds() - start_time
         // petite musique d'alarme toutes les minutes ou toutes les heures
         if elaptime = 60.00
            tone(MUSIC_ALERT,1)
            tone(MUSIC_ALERT,1)
            tone(MUSIC_ALERT,1)
         endif
         if elaptime = 3600.00
            tone(MUSIC_ALERT,2)
            tone(MUSIC_ALERT,2)
            start_time := 0
         endif
         disptime := str(int(elaptime / 60), 2) + ':' + ;
             if(elaptime % 60 < 10, '0' + str(elaptime % 60, 4, 2), ;
                str(elaptime % 60, 5, 2))
         @ stoptop+3, stopleft+10 say disptime
      enddo
      ColorSet(C_STOPWATCH_BOX)
      @ stoptop+3, stopleft+10 say disptime
      key := lastkey()
      if ! (key = 27 .and. ALREADY_ON)
         start_time := 0
      endif
   endif
enddo

// redonne force et vigueur à la touche de fonction annihilée plus haut
// notre premier code block, juste en dessous !!!
if hotkey != 0
  setkey( hotkey, {|p, l, v| popstop(p, l, v)} )
endif

// on remet les choses en place , comme à l'arrivée
restscreen(stoptop, stopleft, stoptop + 5, stopleft + 29, oldscrn)
GFRestEnv()
return NIL

* fin de fonction PopStop()
```

FIGURE N° 6

SUITE FIGURE N°8

```
        * Résultat du passage dans la moulinette du préprocesseur.

        if elaptime = 60.00
           tone(880,1)
           tone(880,1)
           tone(880,1)
        endif
        if elaptime = 3600.00
           tone(880,2)
           tone(880,2)
           start_time := 0
        endif

   disptime := str(int(elaptime / 60), 2) + ":" + if(elaptime % 60 < 10,;
      "0" + str(elaptime % 60, 4, 2), str(elaptime % 60, 5, 2))
   DevPos( stoptop+3, stopleft+10 ) ; DevOut( disptime )
   enddo
   ColorSet(11)
   DevPos( stoptop+3, stopleft+10 ) ; DevOut( disptime )
   key := lastkey()
      if ! (key = 27 .AND. (start_time > 0))
         start_time := 0
      endif
   endif
enddo

if hotkey <> 0
   setkey( hotkey, {|p, l, v| popstop(p, l, v)} )
endif

restscreen(stoptop, stopleft, stoptop + 5, stopleft + 29, oldscrn)
GFRestEnv()
return NIL

static function drawstopbx
local ccolor, ptr
ColorSet(11)
ccolor := setcolor()
shadowbox(stoptop, stopleft, stoptop+6, stopleft+27, 5)

ptr := at("/", ccolor)

setcolor(substr(ccolor, ptr + 1, at(",", ccolor) - ptr - 1) + "/" +;
     substr(ccolor, 1, ptr - 1) )
DevPos( stoptop, stopleft+ 8 ) ; DevOut( "CHRONOMETRE" )
setcolor(ccolor)
DevPos( row()+3, stopleft+ 9 ) ; DevOut( "0:00.00" )
__Box( stoptop+2, stopleft+9, stoptop+4, stopleft+18, "┌┐│┘─└│ " )
return NIL
```

FIGURE N°9

```
  DrawStopBx() -- dessine le boitier du chrono
*/
static function drawstopbx
local ccolor, ptr
ColorSet(C_STOPWATCH_BOX)
ccolor := setcolor()
shadowbox(stoptop, stopleft, stoptop+6, stopleft+27, 5)
// détermine le contraire de la couleur principale
ptr := at('/', ccolor)
setcolor(substr(ccolor, ptr + 1, at(',', ccolor) - ptr - 1) + '/' + ;
        substr(ccolor, 1, ptr - 1) )
@ stoptop, stopleft+ 8 say 'CHRONOMETRE'
setcolor(ccolor)
@ row()+3, stopleft+ 9 say '0:00.00'
SINGLEBOX(stoptop+2, stopleft+9, stoptop+4, stopleft+18)
return NIL

*
```

FIGURE N° 7

```
    * Fichier POPSTOP.PPO

    * Résultat du passage dans la moulinette du préprocesseur.

#line 1 "\grump5\grump.ch"
#line 22 "\grump5\POPSTOP.PRG"
#line 1 "\clipper5\include\inkey.ch"
#line 24 "\grump5\POPSTOP.PRG"

static start_time := 0, stoptop := 4, stopleft := 26

function popstop(gfproc, line, var)

local elaptime, disptime, hotkey := 0, key := 0, tempcolor,;
 oldscrn := savescreen(stoptop, stopleft, stoptop + 7, stopleft + 29)
GFSaveEnv()

if (gfproc <> NIL)
   setkey(hotkey := lastkey(), NIL)
endif

drawstopbx()

while key <> 27
   if ! (start_time > 0)
      DevPos( stoptop+1, stopleft+3 ) ; DevOut( " Alt-F = tâche de Fond
      DevPos( stoptop+5, stopleft+6 ) ; DevOut( "[Espace]= chrono" )
      if (key := ginkey(0)) = -39
         tempcolor := colorpal(tempcolor, 16, if(stopleft > 36, 0, 64))

         if lastkey() <> 27
            ColorSet(11, tempcolor)
            drawstopbx()
         endif
         loop
      endif
      if key < 32 .AND. key <> 27 .AND. key <> 13
         checkmove(key, 5, 29, @stoptop, @stopleft, @oldscrn)
         loop
      endif
   endif
   if key = 289
      start_time := seconds()
      key := 27
   elseif key <> 27 .OR. (start_time > 0)

      DevPos( stoptop+1, stopleft+3 ) ; DevOut( if((start_time > 0),;
         "Chrono en tâche de fond", " ALT-F = tâche de Fond ") )
      DevPos( stoptop+5, stopleft+6 ) ; DevOut( if((start_time > 0),;
        "[Espace]= chrono", "[Espace]= arrêt ") )
      ColorSet(12)
      start_time := if(! (start_time > 0), seconds(), start_time)
      while inkey() = 0
         elaptime := seconds() - start_time
```

FIGURE N° 8

```
/*
        Programme: GRUMP.CH
        Système  : GRUMPFISH LIBRARY
        Auteur   : Greg Lief
        Copyright (c) 1988-90, Greg Lief
        Ceci n'est évidemmment qu'un petit extrait du grump.ch  */

/* des defines simples pour un homme simple (moi, pas vous)
*/
#define OUI          .T.
#define NON          .F.
#define SPACEBAR     32
#define CRLF         CHR(13)+CHR(10)
#define NETERR_MSG  "Erreur réseau,modifications momentanément impossibles"

/* voici un moyen futé pour faire renvoyer par une même fonction
       un tableau et une valeur logique */
#translate loadarray(<array>, <file>) => ;
          (len(<array> := gloadarray(<file>)) > 0)


/*       sapristi, les boites de NANTUCKET ne vident pas l'écran
         donc j'écris les miennes.                              */
#translate SINGLEBOX(<top>, <left>, <bottom>, <right>) => ;
          @ <top>, <left>, <bottom>, <right> BOX "┌┐│┘─└│ "
#translate DOUBLEBOX(<top>, <left>, <bottom>, <right>) => ;
          @ <top>, <left>, <bottom>, <right> BOX '╔╗│╝═╚│ '


/* ;l'incontournable fonction Center() ! */
#translate CENTER(<row>, <msg>) => ;
  DevPos( <row>, int((maxcol()+1 - len(<msg>)) / 2)) ; DevOut( <msg> )

/* Un peu de musique, pour les mélomanes */


#command Charge => tunes({ {523,2}, {698,2}, {880,2}, {1046,4}, ;
                           {880,2}, {1046,8} } )
#command NannyBoo => tunes( { {196,4}, {196,4}, {164,4}, {220,4}, ;
                              {196,8}, {164,8} } )
#command TheFifth => tunes( { {392,2}, {392,2}, {392,2}, {311,8} } )
#command tunes(<a>) => aeval(<a>, { | a | tone(a\[1], a\[2]) } )

/* pendant que nous sommes dans le sujet, quelques constantes ...*/

#define MUSIC_ERROR          440
#define MUSIC_WAITON         {800, 1600}
#define MUSIC_ALERT          880
#define MUSIC_RASPBERRY      50      // insultant mais nécessaire
#define MUSIC_TTY            25      // utilisé dans TTY() et TTY2()
```

*Figure N° 10*

Tél. : 46.03.38.32
Télex : 250752 F
Télécopie : 46.05.28.71

LE MAP

LE MAP
5, Avenue du Maréchal Juin
92100 BOULOGNE

# LOGICIEL ET MAINTENANCE D'APPLICATIONS

## ECRANS

| | |
|---|---|
| — VGA monochrome 12'' plat | 750 F HT ( 890 F TTC) |
| — VGA couleur 14'' | 2 270 F HT (2 690 F TTC) |
| — EGA couleur 14'' | 1 990 F HT (2 360 F TTC) |
| — VGA monochrome 14'' | 835 F HT ( 990 F TTC) |

## IMPRIMANTES

| | |
|---|---|
| — EPSON LQ 1050 | 6 405 F HT (7 597 F TTC) |
| — STAR LC 10 | 1 256 F HT (1 490 F TTC) |
| — BULL 4/24 (134 col., 24 aig.) | 5 173 HT (6 136 F TTC) |
| — IMP LASER 4 p/mn - 1 Mo RAM | 7 890 F HT (9 358 F TTC) |
| — CANON BJ 330 A JET D'ENCRE 136 col. - 300 cps | 4 890 F HT (5 800 F TTC) |

## DISPONIBLE

— LOGICIELS
— CLAVIERS CYRILLIQUES
— TERMINAUX CYRILLIQUES
— CABLES
— DATASWITCH
— CONSOMMABLES DE MARQUE
— Et autres

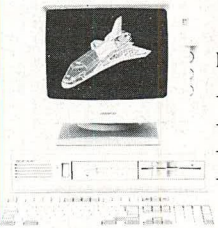
## PROMOTIONS

| | |
|---|---|
| • WINDOWS 2.11 | 290 F HT ( 344 F TTC) |
| • Carte modem LCE TEL 2 | 950 F HT (1 127 F TTC) |
| • Carte VGA 800 × 600 256 Ko (MDA/HERCULES/CGA/EGA/VGA/SVGA) | 790 F HT ( 937 F TTC) |
| • Souris MS série | 900 F HT (1 068 F TTC) |
| • Souris compatible | 190 F HT ( 225 F TTC) |
| • WORD pour WINDOWS | 2 990 F HT (3 548 F TTC) |

Réf. : DL 386 SE

— LAPTOP COMPUTER 80386 SX - 20 MHz - 1 Mo RAM
— FD 1.44 Mo - DD 40 Mo
— 2 × RS 232 - 1 parallèle
— Vidéo VGA
— Clavier détachable avec pavé numérique
— Boîtier extension
— Sac de transport

PRIX LE MAP : 16 490 F HT (19 557 F TTC)



Réf. : CPC 6200

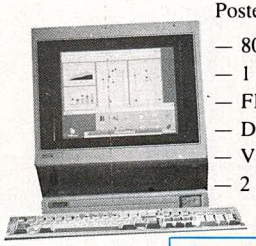
— 80286 / 12 MHz - 1 Mo RAM
— 2 × RS 232 - 1 parallèle
— 2 FD
— Vidéo interface EGA

Option Disque dur : 20 Mo
PRIX LE MAP : 5 990 F HT (7 104 F TTC)
Option Disque dur : 40 Mo
PRIX LE MAP 6 990 F HT (8 290 F TTC)

Réf. : DPC / 386 S

— 80386 SX / 16 MHz - 2 Mo RAM
— Disque dur : 40 Mo - 2 FD
— Vidéo : interface VGA - Port série et parallèle

MAINTENANCE SUR SITE
GRATUITE 1 AN
PRIX LE MAP : 9 990 F HT (11 848 F TTC)



Poste Z - LS (ZENITH)

— 80386 SX / 16 MHz
— 1 Mo extensible à 8 Mo
— FD 1.44 Mo en 3''1/2
— Disque dur : 40 Mo
— Vidéo : VGA
— 2 séries - 1 parallèle

PRIX LE MAP : 12 900 F HT
(15 300 F TTC)

Spécial : Le MAP reprend jusqu'à 10.000 Fm (calculé en fonction du Prix du Portable à acheter) Votre vieil ordinateur pour un portable Zenith.

Serveur 486 (ZENITH)

— Z - 486 / 33 ET
— Intel 486 - 32 bits / 33 MHz - 8 Mo RAM
— Interface VGA
— FD 1.44 Mo en 3''1/2
— Disque dur : 360 Mo - 2 séries - 2 parallèles
— WINDOWS + Souris

PRIX LE MAP : 74 900 F HT (88 831 F TTC)

**La qualité de niveau "Zéro-Défaut", est le prix de la compétitivité de LE MAP**

**LE MAP**

**Revendeur agréé**
**DAEWOO**
**PLUS PRES DE VOUS**

**Revendeur agréé**
**ZENITH**
**data systems**
**Groupe Bull**

# IDFS

**160, AVENUE GENERAL LECLERC, BAT 4**
**B.P. 62 - 91193 GIF SUR YVETTE CEDEX**
**TEL. : 64.46.21.44 FAX : 69.28.07.15**

## MICRO ORDINATEURS

### PC 286 12/16 MHz 1 MO VGA/M

- Lecteurs 1.2 MO et 1.44 MO
- Disque dur 40 MO - 1 Sortie //, 2 Séries
- Carte VGA - Ecran VGA MONO 14"
- Clavier 102 touches - Souris .... 7200 TTC

### PC 386 SX 16 MHz 2 MO VGA/C

- Lecteurs 1.2 MO et 1.44 MO
- Disque dur 40 MO - 1 Sortie //, 2 Séries
- Carte VGA - Ecran VGA COULEUR 14"
- Clavier 102 touches - Souris ....10650 TTC

### PC 386 20/25 MHz 4 MO MTS

- Lecteurs 1.2 MO et 1.44 MO
- Disque dur 40 MO - 1 Sortie //, 2 Séries
- Carte VGA - ECRAN MULTISYNC 14"
- Clavier 102 touches - Souris ....14950 TTC

ET AUSSI :

**COMPAQ**
**HEWLETT PACKARD**
**TANDON**
**TOSHIBA**

### MONITEURS

MONOCHROME BIFREQUENCE

VGA :
MONO & COULEURS
NEC 2A - SONY - TAXAN

MULTISYNC :
NEC : 3D - 4D - 5D
SONY : MULTISCAN

### MEMOIRES

BARETTES SIMM - SIP
CARTES EXTENSIONS MEMOIRES
COPROSSESSURS 287 - 387
MEMOIRES POUR
- COMPAQ - TOSHIBA - VECTRA
RAM DYNAMIQUES
- 41256 - 44256 - 41000 .....

### LOGICIELS

EDUCATIFS :
- CEDIC - NATHAN - WALTDISNEY .....

PROFESSIONNELS :
- ABSOFT - BORLAND - MICROSOFT.....

## SERVICES

### PAO

ARCHIVAGE BANQUE DE DONNEES
SAISIE TRADITIONNELLE
SAISIE OPTIQUE
COMPOSITIONS
MISE EN PAGE
IMPRESSION LASER
IMPRESSION EN OFFSET

### EDITION

LECTORAT
TRAITEMENT DE TEXTE
PREPARATION DE COPIE
RECRYPTAGE ET CORRECTION
MISE EN FORME
CALIBRAGE
PERSONNALISATION DE DOCUMENTS
IMPRESSION
EDITION A COMPTE D'AUTEUR

### CONSEILS

ETUDE ET REALISATION DE PROJET D'EDITION

MISE EN PLACE DE STATION DE SERVICE P.A.O.

CREATION D'UN OUTIL P.A.O.

REALISATION D'OUVRAGES TECHNIQUES ET LITTERAIRES

LOCATION DE MATERIEL

### LOGICIEL P.A.O. & O.C.R.

ART & LETTRE
OMNIPAGE
PAGE MAKER
READY SET GO
VENTURA ....

### RESEAUX NOVELL

INSTALLATION - MAINTENANCE
CABLAGE

### HORRAIRES

**LUNDI AU VENDREDI : 10 H A 18 H**

## PERIPHERIQUES

### IMPRIMANTES LASER

CANON :
- LPB4 - LPB8 III, IIIT, IIIR

HP :
- HP 2P - HP 3, 3D

NEC :
- SW 266
- POSTCRIPT : SW 290, SW2 S60P

**MATERIELS GARANTIES UN AN SUR SITE PAR CONSTRUCTEURS**

### MATRICIELLES

CITIZEN :
- 120 D PLUS - 124 D
- SWIFT 9 / 24

EPSON :
- LX 800 / 850
- LQ 500 / 550 / 850 / .... / 2550

NEC :
- P 20 / 30 / 60 / 70 / 90

PANASONIC :
- KX - P1081 / P1180 / ... / P1695

STAR :
- LC 10 / 15 / 20

### CONSOMMABLES

BUFFERS IMPRIMANTES
CABLES // ET SERIES
CARTOUCHES D'ENCRE
CARTOUCHES POLICES CARACTERES
DISQUETTES 3"1/2 - 5"1/4
PAPIERS LASER (A4) & PHOTOCOPIES
PAPIERS TELECOPIES (FAX)
RUBANS IMPRIMANTES
TONER LASER

### SCANNERS & SOURIS

SOURIS COMPATIBLE MICROSOFT
SCANNER A MAIN
SCANNER GRAND FORMAT (A4 )

### TELECOPIEUR G3

CARTE : PNB - KORTEX

CANON - PANASONIC - TOSHIBA

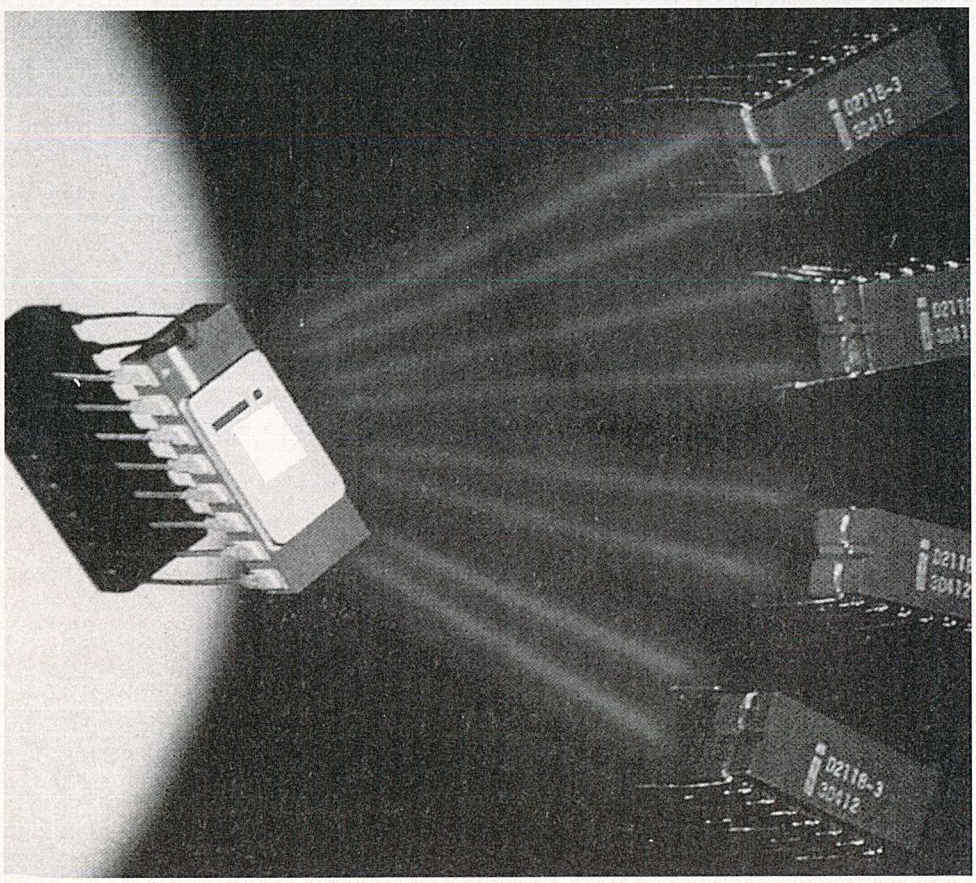
Les marques citées sont déposées

Nos prix indicatifs peuvent être révisés sans préavis. Matériels garantis 1 an pièces et main d'oeuvre, retour en nos ateliers

# Virus, suite... Les virus de partition EDV, OHIO et STONED

**Ce mois-ci, une mécanique infernale pour venir à bout de vos virus de partition, avec le nouveau Turbo C++. Trois virus vont désormais se tenir à carreau : EDV, STONED, et OHIO, pour commencer.**

Jean-Luc NAIL



Ce sera, nous l'espérons, le dernier article d'une trilogie qui a présenté successivement les virus de boot et les virus de fichiers. A la différence d'un Jérusalem [B], ces indésirables s'attaquent à la racine du disque dur, le secteur absolu 0, qui loge une information de la plus haute importance, la table de partition du disque.

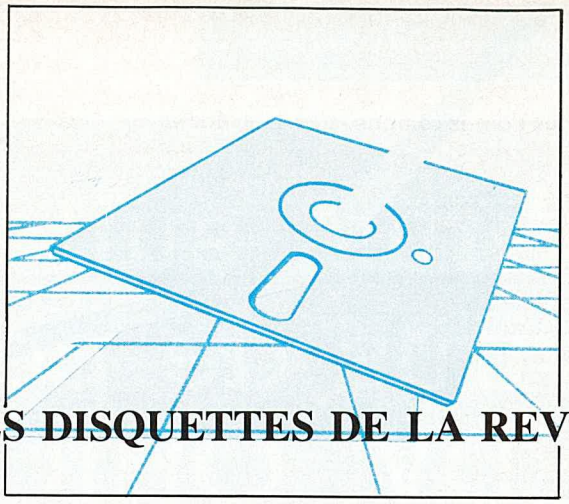
Le courrier des lecteurs l'a mis douloureusement en évidence. Eh oui ! Nous avons reçu de nombreux SOS jusqu'à celui d'un fidèle lecteur de Lomé, au Togo. Cette classe de virus, de toute évidence, fait partie des plus virulentes sur l'hexagone. En terme d'espèces, ils sont assez rares, John Mac Afee, éditeur du maintenant célèbre VIRSCAN [72] de Dennis Yelle, en dénombre seulement une dizaine sur 142 souches de virus identifiées.

Les virus de fichiers sont agaçants, ils empoisonnent à petit feu disques et disquettes, se multipliant à profusion sur les fichiers programmes. Le virus de boot de volume ou celui de partition est unique. Par contre, il s'attaque à la structure même du support sur lequel il s'est agrippé. Sur la disquette, de construction simple et de dimensions connues, la greffe de l'indésirable se déroule, en général, assez bien. Sur les disques durs, la phase de la contamination, plus acrobatique, est elle-même déjà la source de nombreux accidents logiciels. D'où la vigilance indispensable face à ces fauteurs de trouble.

## Secteur 0 ou secteur 0 ?

Sachons bien de quoi nous parlons. Si vous auscultez le secteur 0 d'un disque dur avec l'éditeur de PCTOOLS, c'est le secteur de lancement d'une partition DOS que vous lisez, donc un secteur 0 relatif. Aucun outil se situant dans la logique DOS ne peut accéder au secteur 0 absolu, pour la bonne raison qu'il n'y a pas d'interruption DOS pour cela. C'est le rôle du BIOS et de son interruption 13h souvent abordée dans la Revue de l'Utilisateur PC.

Peu d'utilitaires autorisent le genre d'exercice qui consiste à dépasser les bornes des volumes DOS. Pour notre part, nous utilisons NU.EXE de NORTON 4.5, lancé avec sa précieuse option /M, seul outil qui permettra encore de dialoguer avec un disque dur complétement destroyed après l'éclatement d'une bombe logicielle. Pour les curieux, NORTON NU accepte même de parcourir une partition NOVELL sur le disque dur d'un serveur non dédié, au prix de quelques grognements quand il franchit et visualise certaines zones inaccessible de l'administration du réseau ! N'en disons pas plus.



# LES DISQUETTES DE LA REVUE

| DISQUETTES THÉMATIQUES | | |
|---|---|---|
| N° | Revues | |
| 1 | 10,11 | Simulateur du fonct. d'un micro |
| 2 | 21 à 25 | Debug système et assembleur |
| 3 | 17 | Séquentiel indexé en BASIC |
| 4 | 19 à 23 | Éditeur et DAO en BASIC |
| 5 | 24 à 27 | Étude TURBO PASCAL (jeux, math.) |
| 6 | 28 | Conversion fichier DOS/WORDSTAR |
| 7 | 30, 32 | Étude C et PROLOG |
| 8 | 30 | Jeu SMILEY |
| 9 | 43, 47, 49 | PRSC, XPRINT et param. imprim. |
| 10 | 29, 30, 33, 34 | Imp. & graph., stylescript. |
| 11 | 26-28, 30-37, 38 | Adresses utiles + bib. souris |
| 12 | 52, 53 | Clavier intelligent et PASSWORD |
| 13 | 40, 49 | Scrolling texte + MENUDER |
| 14 | 50, 51 | Fenêtres Pascal + masques saisie |
| 15 | 32, 34, 40 | Hâchurage, Scrolling HGR, XFEN |
| 16 | 48, 54 | Courbes fractales |
| 17 | 39, 41, 42, 43 | Courbes et volumes en 3D |
| 18 | 34, 35 | MINITEL 1 |
| 19 | 36, 37 | MINITEL 2 |
| 20 | 40, 42 | MINITEL 3 |
| 21 | 50, 52, 53 | RT/CMOS, NEWAUX, INT 44H |
| 22 | 46, 48, 49 | MODIF/DEPL, SAUF, BATCH inter. |
| 23 | 38, 39, 41 | Résidents : ZIP5, TABLES, CONTEXT |
| 24 | 42, 43, 44 | Assemb. & DOS (ASM, HEURE, PRSC...) |
| 25 | 52, 53 | Gestion de fichier en TP4 |
| 26 | 40, 45, 54 | DEDICORE, SCAN & DEL, SAUVE |
| 27 | 44, 45, 40 | Étude TP. 4.0 + Fenêtrage HGR |
| 28 | 48 à 54 | Dossier TURBO C |
| 29 | 50, 54 | SECUDOS et prog. invisible |
| 30 | 51, 52, 53, 55 | DAO en TURBO PASCAL |
| 31 | 55, 56 | DBASE |
| 32 | 56, 58, 59, 60... | GENECR : Générateur d'écran en T.P. |
| 33 | 58, 60 | Compil. interp. et désas. PASCAL |

| UTILITAIRES | | |
|---|---|---|
| N° | Revues | Titres |
| 1 | 40 | DEDICORE : Tri sélectif des répertoires pour copier, effacer, renommer |
| 2 | 41 | CONTEXT : Un résident pour connaître votre configuration hard & soft |
| 3 | 43 | PRSC : Un résident pour rediriger et paramétrer la recopie d'écran |
| 4 | 45 | SCAN & DEL : Pour sélectionner et effacer vos fichiers inutiles |
| 5 | 47 | XPRINT : Un programme résident pour configurer l'imprimante |
| 6 | 49 | PARAMIMP : Un résident pour contrôler et utiliser votre imprimante |
| 7 | 54 | LS : Une commande DIR style UNIX |
| 8 | 57 | VDISK : Un disque virtuel dynamique |
| 9 | 57 | COPYDIR : Pour copier vos répertoires sur disquettes |
| 10 | 56 | RESIZE : Optimisez la taille de l'environnement de votre PC |
| 11 | 58 | SOSFILE : Si vous voulez récupérer des fichiers endommagés, cet utilitaire est pour vous |
| 12 | 58 | PCTEST : Pour tout connaître sur la configuration de votre machine |
| 13 | 56, 57, 59 | EXFILE : Analyseur de fichiers PASCAL |
| 14 | 58, 60 | HARDCOPY : Impression graphique d'écran EGA-VGA en N. & B. et nuances de gris |
| 15 | 61 | EDITPAL : Éditeur de palette pour EGA/VGA |
| 16 | 60 | XMEM : Utilisation de la mémoire étendue sous DOS |
| 17 | 64 | ANTIVIRUS : Traitement des virus PingPong et Jérusalem (sources et programmes) |

**BON DE COMMANDE :**
**à adresser à SOFTIN COMMUNICATIONS**
**2 rue d'Amsterdam**
**75009 Paris - Tél. : 42.81.54.27**

Nom .....

Adresse .....

Code Postal ..... Ville .....

Téléphone .....

Veuillez m'envoyer la ou les disquette (s) numéros : THÉMATIQUES (160,00 F) : .....
UTILITAIRES (90,00 F) : .....

Veuillez trouver ci-joint un chèque de..... × 160,00 F soit..... Francs.
..... × 90,00 F soit..... Francs.
Total :..... Francs.

en règlement à l'ordre de Softin Communications.

Date : Signature :

Vous l'aurez compris, c'est le « Master boot sector » qui nous intéresse aujourd'hui. Il y a peu, Daniel Vasseur y a consacré un article très détaillé dans la Revue de l'Utilisateur PC. Les supports n'étant pas partitionnés, le secteur relatif 0 de chargement de volume DOS se retrouve donc de fait au début du disque, mais il ne faut pas le confondre pour autant.

## La contamination...

Inoculer un virus de partition sur sa machine est très simple et rapide. Il suffit de laisser une disquette altérée dans le lecteur A:, porte fermée, et mettre le micro-ordinateur sous-tension. Une saine précaution consistera donc à ne jamais laisser une disquette engagée dans un lecteur sinon pour le transfert des fichiers. Ensuite, le virus ayant pris le contrôle des opérations de lecture écriture, le moindre contact, un simple DIR, sur une disquette saine et non protégée à l'écriture va perpétuer l'épidémie. De micro en micro...

Cela dit, il n'y a aucun obstacle technique à ce que la première souche de l'indésirable soit issue d'un vrai programme EXE. Nous avons la forte impression d'avoir hérité un jour d'un exemplaire de EDV lors de l'installation d'un traitement de texte très connu et protégé à la copie.

Le secteur de partition contient les premiers octets de code du disque que le BIOS va lancer, déterminant ainsi sous quel système d'exploitation la machine va fonctionner. Le code viral va donc s'immiscer dans le processus normal de chargement du système et aura à sa charge la poursuite des opérations pour que tout semble normal.

On peut observer un trait commun de tous ces perturbateurs : comme la mémoire vive basse du micro va être fortement sollicitée par l'installation du DOS, ils vont tenter une translation le plus haut possible en machine. Ils sont donc, en général, très attentifs à un octet ajusté par le POST, période d'évaluation et de test, c'est celui qui précise la taille mémoire disponible, à l'adresse 40:13 hexa.

Il est toujours bon d'avoir un œil sur cette valeur, avec un quelconque utilitaire. Ces virus n'utilisent pas tous la technique du « résident à l'envers », EDV, lui, cherchera à se nicher dans la mémoire des cartes-écran, au-delà de la barre des 640 kilo-octets. Une chose est certaine, les vecteurs de service d'interruption n'ont aucun motif de pointer vers ces parties de la mémoire.

## EDV alias CURSY

Ne consultez pas la VIRLIST de Patricia M. Hoffman, référence absolue dans le domaine viral. Au niveau international, elle qualifiait EDV de rare.


3615 TEASER

*Recevez* ***GRATUITEMENT*** *le logiciel BBT pour télécharger avec votre machine* ***(PC XT/AT - ATARI ST - AMIGA - MAC)*** *et venez prendre nos logiciels du domaine public !*

3615 TEASER

*Plus de* ***10.000 logiciels*** *triés et sélectionnés à votre disposition. Faites votre choix parmi eux. Ils seront chez vous en quelques minutes prêts à l'emploi !*

3615 TEASER

*Notre protocole BBT est un des plus rapides (90 cps) et des plus fiables du marché sous Transpac et nos logiciels sont* ***les meilleurs et les plus récents.***

3615 TEASER

*En quelques minutes chez vous les derniers softs pour* ***PC XT/AT, ATARI ST, AMIGA*** *et* ***MAC*** *: tableurs, traitements de textes, langages, graphisme, musique, section adultes et des jeux par milliers.*

Pour recevoir votre **BBT**, adressez à :

**FRANCE-TEASER**

**22, Grande Rue 92310 SEVRES**

une disquette vierge avec votre nom, prénom, adresse et type d'ordinateur. Joignez 15 francs en timbres pour frais d'expédition. Vous le recevrez sous 48 h.

*Création typon : Club Megaland - 3615 MEGALAND*

Après notre intervention, elle a donc retenu notre contribution pour illustrer son chapitre concernant EDV et admis le pseudonyme dont nous l'avions baptisé, il y a deux ans : CURSY Virus.

Les spécialistes s'accordent pour qualifier EDV de « stealthy virus », ou virus furtif. Ainsi, si le virus est actif en mémoire, la lecture du premier secteur ne retournera pas le code viral, comme il se doit, mais le vrai secteur de boot ! L'astuce est simple mais imparable, y compris des sniffers de virus les plus affutés, s'ils ne respirent pas la mémoire vive de la machine pour constater qu'il y niche un EDV.

Ce virus est dangereux. Quand sa condition de déclenchement est rencontrée, la prise de contact avec six disquettes saines au cours d'une même session, il reformate les premières pistes de tous les disques qu'il rencontre. Le clavier est alors inhibé, excluant tout redémarrage à chaud de la machine, et le message « That rings a bell, no ? From CURSY » apparaît à l'écran. On ne trouvera pas cette chaîne de caractères dans le code viral, ni sur disque, ni en RAM, elle est cryptée à base de OU exclusif et seulement décodée à l'activation de la bombe.

## OHIO

OHIO possède une caractéristique qui le rendrait presque attachant : il se conduit comme un antivirus du tueur PAKISTANI BRAIN, apparu dans un trafic véreux de copies illiégales de LOTUS 123, en provenance du Pakistan, il y a cinq ans. OHIO guérit les disquettes altétées par BRAIN ! Mieux encore, par le jeu des chaînes d'identification, il assurera une vaccination à toute nouvelle tentative d'agression de BRAIN.

Chaque virus ayant son lot de finesse de programmation, OHIO n'échappe pas à la règle. Son code résiste au redémarrage à chaud de la machine (CTRL-ALT-DEL) par un détour de l'interruption clavier. Le vrai redémarrage, quant à lui, se fera par le curieux CTRL-ALT-F5.

## STONED

Dans la version que nous a fournie un lecteur de la Revue de l'Utilisateur PC, c'est l'archétype du virus triste, pas méchant pour deux sous, qui ne fait qu'écrire bêtement le message « Your PC is now stoned » à l'allumage du micro. Très facile à capturer, il pourra servir de victime à vos premiers désassemblages.

## Le programme RELIVE.CPP

Un avertissement préalable, le programme de ce mois se voudra plutôt démonstratif des vertus d'un nouveau langage. En quelques centaines de lignes, il n'a pas l'ambition de rivaliser avec la multitude d'antivirus de la distribution share-ware ou commerciale. Entre autres, un point très important, il n'assure pas la détection en mémoire d'un de ces éventuels empêcheurs de tourner en rond.

Quelle stratégie adopter face à un virus parasite de table de partition ? La plus délicate et complexe, que savent faire NORTON DISK DOCTOR ou son homologue du package PC TOOLS 6.0, consisterait à reconstruire la table de partition après analyse des restes du disque dur ? Non, il y a plus simple : savoir comment agit le virus à son installation, cela demandera bien sûr sa capture et son analyse. Il suffit de fourbir son désassembleur... Nous ne recommencerons pas notre ritournelle.

Le passage à un langage orienté objet (POO) amènera quelques perturbations dans les habitudes de travail. Plus question de se jeter sur le clavier sans avoir bien réfléchi au résultat escompté. La phase de l'analyse du problème va donc être prédominante par rapport à l'écriture même du programme, ce n'est que justice rendue.

Dans le cas présent, de quoi avons-nous besoin ? Nous serons amenés à refaire le travail inverse de celui du virus lors de sa greffe sur le disque, d'où des lectures et écritures de secteurs absolus et la nécessité d'une classe clDiskData d'objets « tranche de disque ». Un objet de ce type devra contenir une zone de données pour ses échanges. Il était possible à cette fin d'utiliser le tas. Plus élégant et rentable consistera à créer clTampon, une classe d'objets « tampon mémoire », dynamique et plus à même de contrôler des débordements éventuels.

Ensuite, nous devrons faire des mouvements de données entre ces tampons et parties de disque. La surcharge de l'injecteur >> nous fournira un excellent symbolisme, proche du graphisme pour la rédaction de la partie traitement de chacun des virus.

Le secteur de partition possède un statut lié à sa position géographique sur le disque. En découle la classe clBoot et si l'on suppose que ce secteur est altéré, on le décrira grâce à une dernière classe, clVirusBoot. Cette classe dérivée hérite de toutes les méthodes de ses parentes, plus la faculté propre de témoigner d'une altération quelconque.

## Concepts POO mis en œuvre dans le programme

Sans vouloir paraître académique, passons en revue les notions nouvelles abordées ici par rapport à une programmation classique. On ne pourra pas en quelques lignes développer, ce qui demande les livres entiers qui mobilisent les étagères des librairies spécialisées. Signalons au passage que les documentations Borland sont de plus en plus didactiques et passionnantes, bien plus pédagogiques que nombre des ouvrages précités. Prévoir aussi une bonne étagère pour poser les neuf tomes du package professionnel.

### Classes et héritage de classe

Le principe même de la POO, c'est la propriété d'habiller des données avec les fonctions de traitement qui s'y rattachent. De plus, Turbo C++ autorise l'héritage multiple, c'est-à-dire la faculté de réaliser des objets logiciels à partir de plusieurs autres.

Avec le compilateur Borland, nous avons dans les mains un véritable jeu de LEGO. L'association de ces briques logicielles va nous permettre de construire de toutes pièces des structures très élaborées, qui plus est, dotées d'une « intelligence » par les méthodes qu'elles contiennent.

### Polymorphisme des fonctions

L'informatique aime les noms barbares. Celui-ci cache tout simplement l'idée que l'on puisse, dans un objet dérivé, redéfinir une fonction déjà déclarée dans une classe de la hiérarchie dont a hérité cet objet, ceci avec ou sans le même argumentaire. Exemple : clBoot::Infos(void) n'a pas le même comportement que sa parente clDiskData::Infos(void). La seule condition concerne le type de l'argument de retour de cette fonction, que l'on ne pourra changer.

### Arguments de fonction par défaut

La possibilité d'admettre des valeurs par défaut à l'argumentaire des fonctions n'est pas typiquement un concept de POO, mais procure une facilité d'écriture de plus dans la rédaction de ces fonctions.

### Utilisation d'un objet comme variable d'un autre objet

La classe clDiskData stockera les informations échangées avec le disque dans un objet interne Buffer de classe clTampon. Se pose donc l'initialisation de Buffer lors de la déclaration d'un objet de classe clDiskData, sachant que la dimension de Buffer n'est pas connue à l'avance.

### Initialisation ascendante de la hiérarchie des classes et des objets internes

Dans une hiérarchie d'objets, la construction commence par la classe de base, la plus haute de la dépendance et chaque classe héritée sollicitera son constructeur pour aboutir à l'objet final de la pyramide. Il peut être nécessaire, comme c'est le cas dans ce programme de forcer des valeurs d'initialisation dans le sens inverse. Turbo C++ autorise ce mécanisme.

Utilisation du flux standard cout

Les affichages emploient l'objet « cout » (prononcer c-aout) qui traitera de données formatées décimales et héxadécimales avec contrôle de la largeur de champ. Plus de printf() scabreux.

Surcharge des injecteurs de flux pour des types utilisateur

Le C, qu'on qualifie à juste titre de langage orienté système ignore le composant de base des machines, l'octet ! Chaque programme C doit donc définir les types byte et word. Et de façon à les éditer correctement, on devra, dorénavant redéfinir les méthodes des flux standards pour traiter de ces types.

Gestion de variables sur le tas avec les procédures standards du C++ new et delete

La classe tampon mémoire clTampon ne connaît pas ses mensurations à la compilation, elle sera donc construite sur le tas. Les procédures standards du C, malloc() et free() sont remplacées par les nouvelles fonctions de gestion d'objets sur le tas.

Surcharge d'opérateurs pour les classes du programme

Pour l'exercice, deux types de déclaration équivalents sont utilisés : opérateur inline et opérateur ami. Cette surcharge, concept génial du C++ issu de ADA, utilise la notion un peu troublante de référence de type. Vous trouverez également dans ce programme des surcharges des opérateurs crochets et parenthèses.

## En guise de conclusion...

Au prix d'une préparation et d'une écriture plus délicate, ce langage, nouveau pour la plupart d'entre nous, va bouleverser nos habitudes. Qu'allons-nous y gagner ?

La manipulation d'entités nettement plus complexes, ceci avec un symbolisme beaucoup plus sobre grâce à la surcharge des opérateurs. Avouez que la séquence vaut son pesant de cacahuètes.

```
...
VraiBoot.Lire();
  // passage dans le tampon mémoire
VraiBoot >> Tampon;
... modifications du tampon ...
  // puis l'on pousse le tampon mémoire
  // vers le futur secteur de boot
VirusBoot << Tampon;
  // écriture sur le disque
VirusBoot.Ecrire();
...
```


# Quatrièmes Assises de l'Intelligence Artificielle et de la Formation

## les 18 et 19 Juin 1991
## Palais des Congrès
## Porte Maillot Paris 17e

**L'objectif des Quatrièmes Assises de l'Intelligence Artificielle et de la Formation est de faire le point annuel des applications ou expériences ayant déjà fonctionné.**

**Il existe, dans ce domaine, une grande variété de recherches et de réalisations, du fait notamment que l'Intelligence Artificielle et les Systèmes Experts apportent des solutions à la Formation, à l'Enseignement Assisté par Ordinateur, (EIAO), ou à la Formation Multimédia.**

**Ce sont, notamment : les systèmes d'EIAO dits "intelligents" et Systèmes d'Aide à la Décision et au "Management" servant de support à la formation, et les tuteurs intelligents.**

**La 1re journée fait le point des outils de l'Intelligence Artificielle pour la Formation, les systèmes d'EIAO ou de Formation Multimédia, faisant appel à l'Intelligence Artificielle (Hypermédias, Langages Orientés Objet).**

**La 2e journée met l'accent sur les applications et montrer leurs apports opérationnels dans une démarche de la formation, et les problèmes d'évaluation qui leur sont liés.**

**Renseignements, programmes et inscriptions**
**Journal de la Formation Continue et de l'E.A.O.**
**2, rue d'Amsterdam**
**75009 Paris**
**Tél. : 47.64.07.57. - 42.67.93.80.**

Une meilleure capitalisation des programmes antérieurs. La réutilisation du code est infiniment plus sûre qu'avec le C classique. L'emboîtage et la cohabitation de divers modules, en fait, l'étape de la construction du programme, donnera la sensation d'une souplesse inconnue jusqu'alors.

Les premiers abords d'une programmation C++ seront frustants, si l'on joue la règle du jeu. Chaque entité pouvant avoir un des trois statuts suivants : private, protected et public, cette règle consiste à tout cacher, puis à ne découvrir que le nécessaire.

Un écueil à éviter, pensons-nous, sera de continuer à réfléchir en C dans l'environnement C++. Le passage de l'un à l'autre est si facile que c'est en même temps une force et une faiblesse pour ce nouveau langage.

Et sans aller jusqu'à jeter vos anciens compilateurs aux orties, laissez-vous séduire par C++ !.

***Bibliographie***

*1) « Techniques de traitement d'un virus, le virus PingPong », Jean-Luc NAIL, n° 64 de la Revue de l'Utilisateur PC, Septembre 1990.*

*2) « Les virus squatters de fichiers », Jean-Luc NAIL et Patrick GARDETTE, n° 66 de la Revue de l'Utilisateur PC, Novembre 1990.*

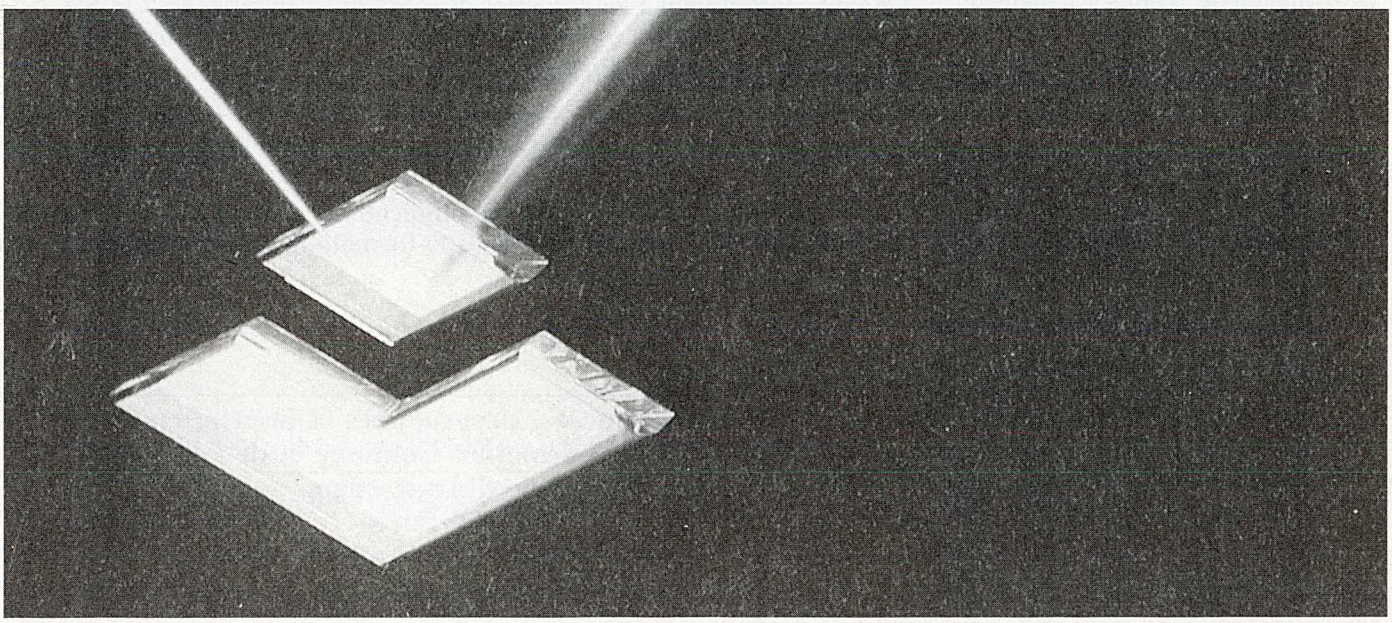
*3) « L'air des partitions », Daniel VASSEUR, n° 68 de la Revue de l'Utilisateur PC, Janvier-Février 1991.*

[annexe]

## Extrait de la VIRLIST de P.M. HOFFMAN

La réputation de ce document n'est plus à faire, c'est la référence absolue de tous les spécialistes en la matière. Une nouvelle édition paraît chaque mois, la livraison de janvier 1991 fait à elle seule presque deux cents pages. Pour en trouver les références, consultez les anciens numéros de la Revue.

```
Virus Name:  EDV
 Aliases:     Cursy, Stealth Virus
 V Status:    Rare
 Discovered:  1988
 Symptoms:    BSC; partition table corruption; unusual system crashes
 Origin:      France
 Eff Length:  N/A
 Type Code:   BRX - Resident Boot Sector/Partition Table Infector
 Detection Method:  ViruScan V58+, IBM Scan, Pro-Scan 1.4+
 Removal Instructions:  MDisk/P, CleanUp V67+, or Pro-Scan 1.4+
 General Comments:
       The EDV, or Cursy, Virus was first discovered in Le Havre, France in
       1988 by Jean-Luc Nail.  At that time, it was named the Cursy Virus.
       Later, in January 1990, it was isolated separately and named the
       EDV virus.  This virus is a memory resident infector of floppy
     .../...

     .../...
       boot sector on infected floppy disks and the partition table of
       infected hard drives, though it cannot be seen if the virus is
       in memory:

            «MSDOS Vers. E.D.V.»

       Jean-Luc Nail has indicated that the EDV or Cursy virus is quiet
       common in the Le Havre area of France, although it is rare outside
       of France.
```



[annexe]

## Une méthode infaillible (?) pour traiter des virus

Si l'on suppose la présence d'un virus sur son micro-ordinateur, la démarche pour le traiter s'avère être très simple et se résume à ces quelques manipulations:

1) Eteindre le micro-ordinateur. Nous l'avons vu avec OHIO, relancer la machine sans l'éteindre ne purge pas toujours la mémoire de portions de code quelque peu malignes.

2) Garder son sang-froid et rassembler les outils nécessaires qui se réduisent à peu de choses. Il nous faudra une disquette système réputée saine, c'est le cas si le disque n'a pas d'encoche de protection. Deuxième outil indispensable, un bon sniffer de virus, dans une version récente. Pour notre part, nous utilisons avec succès celui de Mac Afee, livré en deux parties, le détecteur sous le nom SCAN.EXE et l'éradiqueur CLEAN.EXE.

3) Rallumer la machine avec la disquette système et lancer le sniffer de virus. Au besoin, ordonner l'examen de la mémoire vive, deux précautions valant mieux qu'une dans le domaine. Prendre du recul par rapport à l'indication effective du détecteur. Quelquefois, il se trompera sur l'identification exacte du parasite, en particulier avec les virus fortement cryptés comme la série des [CASCADE/17xx] qu'il confondra facilement avec un quelconque [JOJO]. Mais c'est sans grande importance, il s'agit alors de virus de même parenté et la procédure d'éradication sera la même.

4) Si d'aventure, il y a effectivement un locataire sur le disque, (à propos, pour qui l'ignorerait encore, les virus du monde PC logent toujours sur les disques et disquettes , passer au nettoyage. La plupart du temps, CLEAN suffira. Plus rarement, il faudra user d'un logiciel plus spécialisé tel que M-DISK, du même distributeur. Recommencer les opérations jusqu'à ce que le sniffer de virus se soit complétement calmé. Les cas d'infections multiples sont courants dans les milieux fortement exposés...

Dernière mesure, si votre machine a été phagocytée, passer TOUTES vos disquettes au même dépistage systématique.


# Bulletin d'abonnement pour 1 an

**10 numéros au prix de : France : 220 F T.T.C., Etranger : 300 F**
**Je souscris un abonnement d'un an à la Revue de L'utilisateur de l'IBM-PC**

Nom : .....
Société : .....
Adresse : .....
.....
Code postal : ..... Ville : ..... Tél. : .....

Chèque bancaire joint à l'ordre de SOFtin Communications (France).

**A renvoyer à : La Revue de l'Utilisateur de l'IBM—PC, 2, rue d'Amsterdam, 75009 Paris.**

[annexe]

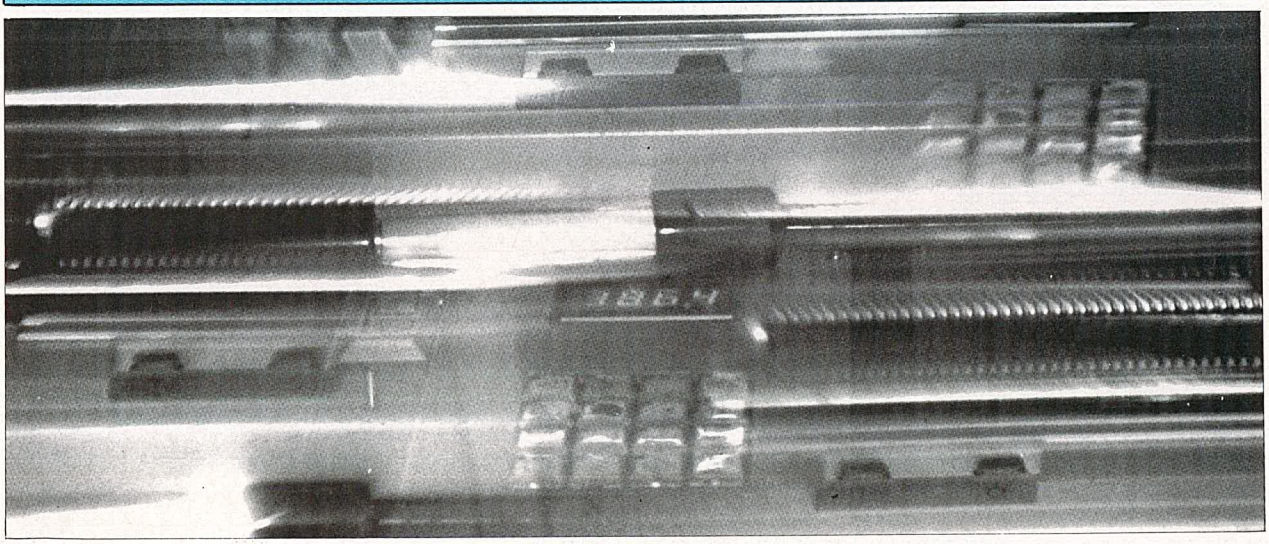
## Quelques octets choisis des virus de partition EDV et OHIO

Découvrir et comprendre les techniques virales contribuent à la démystification du phénomène et à réduire son aspect fantasmatique. C'est avec cet état d'esprit que nous publions des extraits de sources, peu dangereux en soi.

## La technique de camouflage de EDV alias Cursy

Pour les incrédules, moi le premier, voici dévoilée une astuce de EDV qui lui permet d'être invisible quand il est actif en mémoire RAM de la machine. Aucun éditeur de secteur, si perfectionné soit-il, ne le localisera sur les disques. D'où sa réputation de virus furtif. Ces quelques instructions paraissent pourtant bien insignifiantes.

```
; si l'on tente de lire le secteur 0, l'int 13 nous renvoie le secteur de fin de
; disque (pour un format 5 pouces 1/4)
00B0:5A           AbsRead  POP   DX
00B1:59                    POP   CX
00B2:80FE00                CMP   DH,0      ; est-ce la tête 0 ?
00B5:7505                  JNZ   L00BCH    ; saut si non
00B7:83F901                CMP   CX,1      ; est-ce le secteur 1, piste 0?
00BA:741D                  JZ    Sector0   ; saut si oui

; si l'on tente de lire le dernier secteur, l'int 13 nous retourne une erreur de
; commande, c'est son seul système de protection
00BC:81F90827     L00BCH   CMP   CX, 2708H ; est-ce piste 39, secteur 8 ?
00C0:751C                  JNZ   L00DEH
00C2:80FE01                CMP   DH,1      ; tête 1 ?
00C5:7517                  JNZ   L00DEH
.....
00D1:50                    PUSH  AX
00D2:83EC04                SUB   SP,4
00D5:B80001                MOV   AX, 0100H
00D8:CF                    IRET ; retour avec error=»mauvaise commande»

; renvoi du vrai boot, 2 lignes suffisent pour substituer les paramètres
00D9:B601         Sector0  MOV   DH, 1       ; read the last sector
00DB:B90827                MOV   CX, 2708H   ; piste 27 hexa, sect 8
00DE:51           L00DEH   PUSH  CX
00DF:52                    PUSH  DX
00E0:EB41         Sortie1  JMP   SHORT VaLire
00E2:90                    NOP
```



[annexe]

## Une provocation de l'auteur de OHIO ?

Voici, pour qui voudrait le contacter, les coordonnées de l'auteur du virus OHIO telles que l'on peut les découvrir par un affichage de la piste 41 d'une disquette 5 pouces 1/4 altérée. Il a récidivé plus tard avec un second virus de partition, DEN ZUK. Les lettres curieuses (YC1ERP) sont celles de l'identité d'un radio-amateur. Petite colle, essayez vous même de lire le secteur 34, piste 41 d'une disquette 5 pouces 1/4.

```
Cylinder 40, Side 0, Sector 34:

 0000  e9 ed 01 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 0010  20 4e 6f 77 2e 2e 2e 20 59 6f 75 20 41 72 65 20    Now... You Are
 0020  20 20 49 6e 73 69 64 65 20 56 69 72 75 73 20 20     Inside Virus
 0030  20 20 20 20 20 20 20 21 20 20 20 20 20 20 20 20          !
 0040  20 43 4f 4e 47 52 41 54 55 4c 41 54 49 4f 4e 20    CONGRATULATION
 0050  20 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 20    --------------
 0060  20 20 20 56 49 52 55 53 20 31 2e 30 20 20 20 20      VIRUS 1.0
 0070  20 20 28 43 29 20 31 39 38 38 20 62 79 20 20 20     (C) 1988 by
 0080  20 20 54 68 65 20 48 61 63 6b 65 72 73 20 20 20     The Hackers
 0090  20 20 20 20 53 6f 63 69 65 74 79 20 20 20 20 20       Society
 00a0  20 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 20    --------------
 00b0  20 20 59 20 43 20 31 20 45 20 52 20 50 20 20 20     Y C 1 E R P
 00c0  20 30 20 44 20 38 20 37 20 32 20 30 20 39 20 20    0 D 8 7 2 0 9
 00d0  20 44 20 45 20 4e 20 5a 20 55 20 4b 20 4f 20 20    D E N Z U K O
 00e0  20 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 20    --------------
 00f0  20 4a 6c 20 41 6e 63 6f 6c 20 54 69 6d 75 72 20    Jl Ancol Timur
 0100  20 20 20 20 20 58 49 49 2f 31 30 20 20 20 20 20       XII/10
 0110  20 42 61 6e 64 75 6e 67 20 20 34 30 32 35 34 20    Bandung  40254
 0120  20 20 20 49 6e 64 6f 6e 65 73 69 61 20 20 20 20      Indonesia
 0130  20 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 20    --------------
 0140  20 20 20 20 48 61 63 6b 69 6e 27 20 20 20 20 20       Hackin'
 0150  20 20 41 6c 6c 20 54 68 65 20 54 69 6d 65 20 20     All The Time
 ......
```


### Les disquettes de la Revue

## ARTOOL UN LOGICIEL DE D.A.O. POUR PC ET CARTES EGA OU VGA

***ARTOOL possède toutes les fonctions pour exploiter pleinement les possibilités graphiques de votre PC muni d'une carte EGA ou VGA. Il fonctionne en mode BITMAP, et gère toute la palette de couleurs.***

Spécifications techniques :

☐ Fonctionnement en 16 couleurs, 640×350 pour EGA et 640×480 pour VGA.

☐ Dessin à ''main levée'', tracé de droites, de cercles et de boîtes.

☐ Ajustement d'une courbe par la méthode de Bézier .

☐ Fonctions avancées : loupe, miroir, coloriage, texte.

☐ Gestion de la palette, des fichiers images et de l'imprimante. .

ARTOOL fonctionne sur tous compatibles PC muni d'une souris et d'une carte graphique EGA ou VGA. Un minimum de 640 Ko ainsi qu'un disque dur sont vivement recommandés. Des impressions graphiques peuvent être obtenues sur des imprimantes compatibles IBM-GRAPH ou EPSON.

*Fourni avec les sources en langage PASCAL, vous pourrez adapter ARTOOL à vos besoins et à votre configuration. Une aide technique pourra être apportée par nos soins pour l'exploitation de périphériques non prévus dans la version de base (imprimantes, souris, scanners, etc.).*

*ARTOOL est disponible sur disquette 5'' 1/4 (360 Ko), au prix de 450 F (TTC)*

---

BON DE COMMANDE
à adresser à La Revue de l'Utilisateur de l'IBM PC
2 rue d'Amsterdam
75009 Paris - Tél. : 42.81.54.27

Nom .....

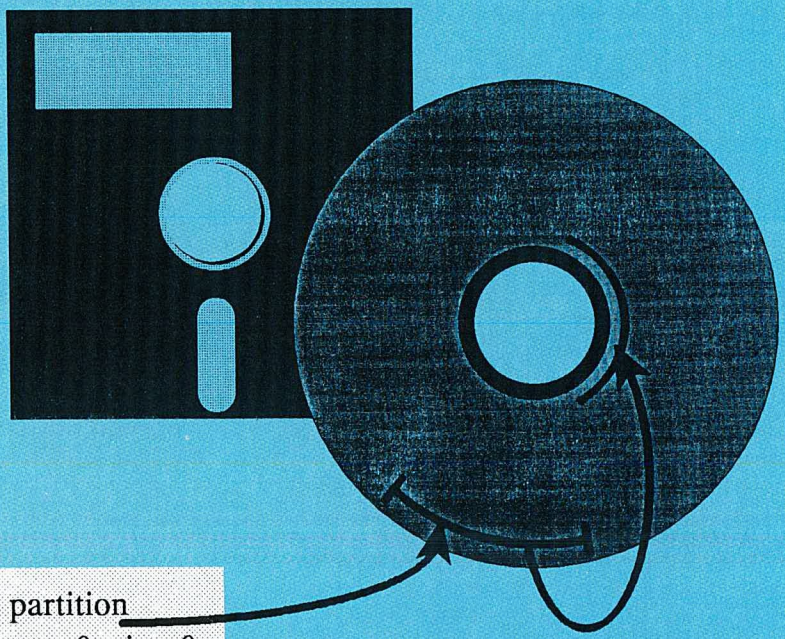
Adresse .....

Code Postal ..... Ville ..... Téléphone .....

Veuillez m'envoyer ..... logiciel(s) ARTOOL, au prix unitaire de 450 F TTC.

Ci-joint un chèque de ..... F. En règlement à l'ordre de Softin Communications.

## Le virus OHIO à l'assaut d'une disquette




### Où se trouve le vrai secteur de lancement du disque ?

La question déterminante pour la rédaction de l'antivirus sera d'analyser la méthode de reproduction du virus. Pour cela, un seul moyen, l'isolation et l'interprétation du code viral, opération s'avérant souvent simple mais quelquefois extrêmement délicate.

### Ohio

Ce virus a adopté le principe de reproduction de Pakistani Brain, qu'il éradiquera s'il le détecte. Sur une disquette 5 pouces 1/4, il formate donc une 41ème piste, au delà du formatage légal, avec des n° de secteurs également illégaux, de 33 à 38. Le secteur 33 recueille le vrai secteur de boot, les cinq autres secteurs reçoivent la suite du code viral.

### EDV alias CURSY

Ce virus repousse le vrai secteur 0 vers ce qui représente le dernier cluster d'une disquette 5 pouces 1/4. S'il s'attaque au disque dur, il ne fera pas de différence, la table de partition se trouvera repoussée piste 39, secteur 8, tête 1, position on ne peut plus catastrophique, puisqu'au milieu des premiers fichiers du volume. Cette procédure d'installation se révèle être extrêmement dangereuse.

### Stoned

Stoned a adopté une méthode encore différente. Si le support est une disquette, le vrai secteur de lancement sera reporté au supposé dernier secteur du répertoire principal. Pari relativement sûr en format 5 pouces, très facheux pour une disquette 3 pouces. De plus, il impose à cette disquette la table de "description de média" de sa jumelle de taille supérieure et réduit donc sa capacité à 360 kilo-octets. Génant. Si le disque est dur, la manoeuvre est plus douce: la table de partition est reculée dans la zone franche de la piste 0, secteur 7, donc relativement moins vulnérable.

```
//-------------------------------------------------------------------//
//                                                                   //
//      Mécanique de traitement de la classe des virus de boot       //
//                                                                   //
//     Jean-Luc NAIL et la Revue de l'Utilisateur de l'IBM-PC        //
//                                                                   //
//     11 février 1991                                               //
//                                                                   //
//-------------------------------------------------------------------//

/*

hiérarchie des classes: clDiskData -> clBoot -> clVirusBoot
                        clTampon
                        clEquipment
clTampon:
  definit un tampon mémoire et ses méthodes d'accès
  La classe clDiskData possède un objet de ce type, sans en dépendre
  hiérarchiquement.

clDiskData:
  définit une portion absolue de disque dur ou souple en mémoire,
  en termes de piste, tête, secteur et longueur
  les fonctions attachées permettent la lecture, l'écriture
  la modification et l'accès contrôlé.
  A sa construction, elle a la charge d'initialiser l'objet clTampon
  qu'elle contient.

clBoot:
  est un cas particulier de sa classe parente, c'est le premier
  secteur absolu d'un disque (ne pas confondre disque et volume).
  Cette classe s'initialise avec un N° de disque et provoque l'initialisation
  de sa classe parente avec la position géographique du secteur 0.
  Une méthode permet de visualiser la «table de description du média» qu'il
  est supposé contenir.

clVirusBoot:
  représente le cas particulier d'un secteur 0 que l'on suppose altéré.
  Ses méthodes permettent donc, en plus des méthodes héritées de tester
  l'altération de cet objet.
  L'identificateur ID de l'altération est redéfinissable.

clEquipment:
  classe pleine d'avenir, chargée de répondre aux questions que l'on se
  posera sur le matériel. Ici, elle répond du nombre de disques en ligne
*/

// forçage du modèle small
#pragma option -ms
#ifndef __cplusplus
      #error Vive le C++ !!!
#endif

// fichiers include typiquement C++
// #include «iostream.h» (appelé par fstream.h)
#include «fstream.h»
#include «iomanip.h»

#include «conio.h»
#include «ctype.h»
```

**RELIVE.CPP Page 1**

```
#include «dos.h»
#include «mem.h»
#include «stdlib.h»
#include «stdio.h»
#include «string.h»

#define SECTOR 0x200
#ifndef TRUE
  #define TRUE      1
  #define FALSE     0
#endif
#define DOS_ID   0xAA55
#define Break() (getch()==27)

// signatures des virus de boot connus

#define OHIO_ID  «\xfa\xfa»     // signature de OHIO utilisée par lui-même
#define OHIO_OF  0x2b           // peut-être un peu courte
#define JMP_OHIO «\xeb\x29»

#define EDV_ID «MSDOS Vers. E.D.V.»
#define EDV_OF 0x1ea

#define STONED_ID «LEGALISE»
#define STONED_OF 0x1a5
//--------

typedef unsigned char byte;
typedef unsigned int word;
typedef unsigned int bool;

// surcharge de l'injecteur de flux <<
// pour les types non prédéfinis byte et word

ostream& operator<< (ostream& Flux, byte B) {
  return Flux << (unsigned long) B;
}

ostream& operator<< (ostream& Flux, word W) {
  return Flux << (unsigned long) W;
}

// prédéclaration
class clDiskData;

//-------------------------------------------------------------------//
//                          CLASSE   clTampon                        //
//-------------------------------------------------------------------//
class clTampon {

protected:
  word Taille;              // en bytes
  byte * Adresse;
  bool IsValid;          // booléen de validité du contenu

public:
  clTampon (byte);
  ~clTampon (void);
  void Validation(void) { IsValid = TRUE; }
  byte * Contenu (void) { return Adresse; }
```

**RELIVE.CPP Page 2**

```
  void Fill (byte Byte) { memset (Adresse, Byte, Taille); }
  void Clear (void) { Fill (0); }
  void Remplir (byte * Byteptr) { memcpy (Adresse, Byteptr, Taille); }
  void Remplir (clDiskData& DD); // défini après la classe clDiskData
  // Remplir admettra comme argument donc soit un pointeur de byte
  // soit une référence à un objet de classe clDiskData

  istream& Read(istream&);// lecture à partir d'un fichier
  ostream& Write(ostream&);      // écriture vers un fichier

  // surcharge de l'opérateur crochet, (on pourrait tester les bornes)
  byte& operator [] (word Offset) { return Contenu () [Offset]; }

  friend ostream& operator << (ostream&, clTampon&);

} ;

//----------------------------------------

// constructeur de la classe clTampon, par défaut 1 secteur
clTampon::clTampon (byte NombreSecteur = 1) {
      Taille = (word) SECTOR * NombreSecteur;
      // réservation sur le tas
      Adresse = (byte *) new char[Taille];
      Clear();
      IsValid = FALSE;
      if (!Adresse) {
          cout << endl << «Mémoire disponible insuffisante pour clTampon»;
          exit (1);
      }
}
// destructeur
clTampon::~clTampon (void) { delete Adresse; }

istream& clTampon::Read(istream& InFile) {
  InFile.read(Adresse, Taille);
  // on devra gérer l'erreur
  return InFile;
}

// écriture vers un flux
ostream& clTampon::Write(ostream& Stream) {
  Stream.write(Adresse, Taille);
  // on devra gérer l'erreur
  return Stream ;
}

// équivalent: surcharge de l'injecteur de flux ostream
ostream& operator << (ostream& Stream, clTampon& Tampon) {
  Stream.write((unsigned char *) Tampon.Contenu (), Tampon.Taille);
  // on devra gérer l'erreur
  return Stream;
}

//-------------------------------------------------------------------------//
//              CLASSE    clDiskData                                       //
//-------------------------------------------------------------------------//

class clDiskData {
```



```
protected:
  byte Head, Disk, Sector;
  word Track;              // disque et position physique
  byte * Base, Size,                  // taille en secteurs
       LastError;              // dernière erreur BIOS
  bool IsValid,            // témoin de la lecture du disque, tampon valide
       WriteEnable, // autorisation d'écrire sans question
       DumpEnable;  // autorisation d'affichage
       // les indicateurs précédents pourront être validé par option
       // de la ligne de commande

public:
  clTampon Buffer;               // objet interne de classe clTampon
  // l'accès (public) aux méthodes de cet objet est simple
  // exemple: VirusBoot.Buffer.Fill(0);

  clDiskData (word Piste, byte Secteur, byte Taille, byte Disque, byte Tete);
  void Infos (void);
  word Lire  (void);              // du disque
  word Ecrire (void);      // vers le disque
  word Afficher (void);           // vers l'écran
  void Modifier (byte * Tampon) { memcpy(Base, Tampon, Size * SECTOR); }
  void Recopier (clDiskData DD) { Modifier(DD.Base); }
  byte * Contenu (void) { return Base; }
  word Taille (void) { return Size * SECTOR; }
  bool IsDos (void);
  void Bavard (void) { DumpEnable = TRUE; }
  // option d'affichage par défaut
  void Muet (void) {  DumpEnable = FALSE; }

  // injecteurs définis en fonctions inline
  // passage d'un seul argument, l'opérateur s'applique à «this», sous-jacent
  clDiskData& operator << (byte *);
  byte *      operator >> (byte *);
  clDiskData& operator << (clTampon&);
  clTampon&   operator >> (clTampon&);

  // injecteurs définis en fonctions amies, passage de deux arguments
  friend clDiskData& operator << (clDiskData&, clDiskData&);
  friend clDiskData& operator >> (clDiskData&, clDiskData&);
} ;

//-------------------------------------------
// opérateurs associés à la classe clDiskData
//-------------------------------------------

inline clDiskData& clDiskData::operator << (byte* Tamp) {
// usage: DiskData << Tampon;
// avec contrôle du débordement
  this->Modifier(Tamp);
  return *this;
}

inline byte * clDiskData::operator >> (byte* Tamp) {
// usage: DiskData >> Tampon;
// attention, possibilité de débordement
  memcpy (Tamp, this->Base, this->Taille());
  return Tamp;
}

inline clDiskData& clDiskData::operator << (clTampon& Tampon) {
```

```
// usage: DiskData << Tampon;
// avec contrôle du débordement
  this->Modifier(Tampon.Contenu());
  return *this;
}

inline clTampon& clDiskData::operator >> (clTampon& Tampon) {
// usage: DiskData >> Tampon;
// contrôle du débordement
  Tampon.Remplir(this->Base);    // par le pointeur de byte
//  Tampon.Remplir(this); // synonyme par l'objet
  return Tampon;
}

// opérateurs fonctions, amis de la classe clDiskData

clDiskData& operator << (clDiskData& DD1, clDiskData& DD2) {
// usage: DiskData1 << DiskData2;
// contrôle du débordement
  DD1.Modifier (DD2.Contenu());
  return DD1;
}

clDiskData& operator >> (clDiskData& DD1, clDiskData& DD2) {
// usage: DiskData1 >> DiskData2;
// contrôle du débordement
  DD2.Modifier (DD1.Contenu());
  return DD2;
}

//-------------------------------------------
// fonctions associées à la classe clDiskData
//-------------------------------------------

// constructeur de la classe clDiskData avec son argumentaire par défaut
// sa 1ère tâche consiste à initialiser son objet interne Buffer de
// type clTampon
clDiskData::clDiskData (word Piste = 0, byte Secteur = 1,
                        byte NbreSecteur = 1, byte Disque = 0,
                        byte Tete = 0) : Buffer (NbreSecteur) {
  Track = Piste;
  Sector = Secteur;
  Disk = Disque;
  Head = Tete;
  Size = NbreSecteur;
  LastError = WriteEnable = IsValid = DumpEnable = FALSE;
  // récupération de l'adresse du tampon
  Base = Buffer.Contenu();
  if (!Base) {
       cout << endl
            << «Pas assez de mémoire pour un objet de classe clDiskData»;
       exit(1);
  }
} ;

word clDiskData::Lire(void) {
  struct REGPACK Register;
  byte far * Adresse;
  word Retry = 5, Error = 1;

  Adresse = (byte far *) Base;
```



```
  while (Retry && Error) {
       /* reset disk system */
       Register.r_ax = 0;
       intr(0x13, &Register);
       Register.r_ax = 0x0200 + Size;
       // offset du tampon Adresse
       Register.r_bx = (word) (((unsigned long) Adresse) & 0x0000ffff);
       // pour les disques durs, les 2 bits hauts de CL sont
       // les 2 bits hauts de la piste, codées sur 10 bits
       Register.r_cx = Sector + ((Track & 0x0300) >> 2) + (Track << 8) ;
       Register.r_dx = Disk    + (Head << 8);
       //segment du tampon Adresse
       Register.r_es = (word) (((unsigned long) Adresse) >> 16);
       intr(0x13, &Register);
       /* renvoie le code de l'erreur éventuelle, carry == 1 */
       Error = (Register.r_ax >> 8);
       Retry--;
  }
  IsValid = TRUE;   // la classe pourra savoir si le tampon a été chargé
  Buffer.Validation();
  return LastError = Error;
}

word clDiskData::Ecrire(void) {
  struct REGPACK Register;
  byte far * Adresse;      // adresse de format long à passer au DOS
  word Retry = 1, Error = 1;      // par défaut un seul essai
  Adresse = (byte far *) Base;
  if (!WriteEnable) {
       char c = 0;
       cout << endl << «Ecriture sur le disque ? (O/N) «;
       cin >> c;
       if (toupper(c) != ''O'') return TRUE;
  }
  // on pourra aussi tester IsValid
  while (Retry && Error) {
       /* reset disk system */
       Register.r_ax = 0;
       intr(0x13, &Register);
       /* piste 40, secteur 33 */
       Register.r_ax = 0x0300 + Size;
       Register.r_bx = (word) (((unsigned long) Adresse) & 0x0000ffff);
       // pour les disques durs, les 2 bits hauts de CL sont
       // les 2 bits hauts de la piste, codées sur 10 bits
       Register.r_cx = Sector + ((Track & 0x0300) >> 2) + (Track << 8) ;
       Register.r_dx = Disk    + (Head << 8);
       Register.r_es = (word) (((unsigned long) Adresse) >> 16);
       intr(0x13, &Register);
       /* renvoie le code de l'erreur éventuelle, carry == 1 */
       Error = (Register.r_ax >> 8);
       Retry--;
  }
  return LastError = Error;
}

word clDiskData::Afficher(void) {
  int i, j, k;
  if (!DumpEnable) return TRUE;

  /* séquence affichage hexa et ASCII */
  for (i = 0; i < Size * SECTOR; i += SECTOR) {
```

```
      cout << «\n\nPiste « << dec << Track
           << « / Secteur n° « << ((i >> 9) + Sector);
      cout.fill(''0'');          // complète l'hexa avec des 0
      cout << hex;
      cout.setf(ios::uppercase);  // hexa en majuscules

      for (j = 0; j < SECTOR; j += 0x10) {
            /* point d'arrêt et de reprise sur appui clavier */
            if (kbhit()) {
                  if (Break()) return FALSE;
                  if (Break()) return FALSE;
            }
            // affichage de l'offset de début de ligne
            cout << endl << setw(4) << (j + i) << « : «;

            // affichage du corps de la ligne en hexadécimal
            for (k=0; k<0x10; cout << setw(2) << Base[j+j+k++] << '' '');

            cout << «: «;

            // affichage de la partie ASCII
            for (k = 0; k < 0x10; k++)
              cout << (char) (Base[i + j + k] >= '' '' ?
                        Base[i + j + k] : ''_'');
      }
  }
  cout << endl;
  return TRUE;
}

void clDiskData::Infos(void) {
  cout << «\n\nInfos sur l'initialisation de l'objet clDiskData:»
       << «\nTête    « << dec << Head  << «\nSecteur « << Sector
       << «\nPiste   « << Track << «\nTaille  « << Size;
}

bool clDiskData::IsDos(void) {
  return (Base[0x1fe] == (DOS_ID & 0x00ff)) && (Base[0x1ff] == (DOS_ID >> 8));
}

//-------------------------------------------------------------------------//
//              CLASSE clBoot                                              //
//-------------------------------------------------------------------------//

class clBoot : public clDiskData {
// pas de variables internes
// clBoot est un cas particulier de clDiskData, avec les méthodes afférentes

 public:
   // le constructeur de clBoot initialise celui de clDiskData
   // avec les paramètres d'un secteur 0 absolu et le disque voulu
   clBoot (byte Disk) : clDiskData (0, 1, 1, Disk, 0) { };
   void Infos(void);
   void InfosDisque(void);
} ;

void clBoot::Infos(void) {
  cout << «\n\nInfos sur l'initialisation de l'objet clBoot:»
       << «\nTête    « << dec   << Head << «\nSecteur « << Sector
       << «\nPiste   « << Track << «\nTaille  « << Size;
}
```



```
void clBoot::InfosDisque(void) {
  struct boot {              // Media Description Table
       byte jump[3],
            OEM[8];
       word Octets_par_Secteur;
       byte Secteurs_par_Cluster;
       word Secteurs_Reserves;
       byte Nombre_de_FAT;
       word Taille_Repertoire,       // en nombre d'entrées (0x20 octets)
            Capacite_Disque;         // nombre de secteurs du volume
                                     // et non du disque
       byte Identificateur_Disque;
       word Taille_FAT,
            Secteurs_par_Piste,
            Nombre_de_Tetes,
            Nombre_Secteurs_Caches;
       byte Garbadge[512 - 32];
       word Signature;             // 0xaa55 pour les intimes
  } ;
  struct boot* Boot;

  Boot = (struct boot *) Base;
  cout << dec << setfill('' '');
  cout << «Table de description du média» << endl
         << «Signature matériel (OEM)    «;
//  printNchar(Boot->OEM, 8);
  cout << «\nNombre d'octets par secteur      «     << setw(3)
         << Boot->Octets_par_Secteur
         << «\nNombre de secteurs par cluster   « << setw(3)
         << Boot->Secteurs_par_Cluster
         << «\nNombre de secteurs par piste     « << setw(3)
         << Boot->Secteurs_par_Piste
         << «\nNombre de têtes de l'unité       « << setw(3)
         << Boot->Nombre_de_Tetes
         << «\nNombre de secteurs cachés        « << setw(3)
         << Boot->Nombre_Secteurs_Caches;
  cout << «\nNombre de secteurs réservés      « << setw(3)
         << Boot->Secteurs_Reserves
         << «\nNombre de F.A.T.                « << setw(3)
         << Boot->Nombre_de_FAT
         << «\nTaille des F.A.T. en secteurs    « << setw(3)
         << Boot->Taille_FAT
         << «\nNombre d'entrées en rép. racine  « << setw(3)
         << Boot->Taille_Repertoire
         << «\nNombre de secteurs du volume     « << Boot->Capacite_Disque;
  cout << «\nTaille du volume en octets    «    << setw(6)
       << ((long) Boot->Capacite_Disque * (long) Boot->Octets_par_Secteur);
  cout << «\nType du disque                «
         << hex << Boot->Identificateur_Disque << dec;
}

//-------------------------------------------------------------------------//
//      CLASSE   clVirusBoot                                               //
//-------------------------------------------------------------------------//

class clVirusBoot : public clBoot {

 protected:
  byte VirusIDstr[80];
  int  VirusIDlen;
  word VirusIDoff;
```

```
  public:
   clVirusBoot (byte);
   // forcer un identificateur de virus
   void VirusID (char *, word, word);

   // surcharge de l'opérateur d'appel de fonction
   clVirusBoot& operator () (char * Str, word Longueur, word Offset) {
        VirusID (Str, Longueur, Offset);
        return *this;
   }
   // voir si le secteur contient le virus désigné
   bool IsVirus(void);
} ;

//------------------------------

// le constructeur de clVirusBoot a la charge d'inialiser sa classe
// parente clBoot avec le disque désiré
clVirusBoot::clVirusBoot (byte Disk) : clBoot(Disk) {
  * VirusIDstr = VirusIDlen = VirusIDoff = 0;
}

void clVirusBoot::VirusID (char * Chaine, word Longueur, word Offset) {
  // tester pour ne pas mettre un ID de plus de 79 caractères
  if (strlen(Chaine) > 79) Chaine[79] = 0;
  strcpy ((char *) VirusIDstr, Chaine);
  VirusIDlen = Longueur;
  VirusIDoff = Offset;
}

bool clVirusBoot::IsVirus(void) {
  int i;
  for (i = 0; i < VirusIDlen; i++)
            if (Base[VirusIDoff + i] != VirusIDstr[i]) return FALSE;
  return TRUE;
}

//---------------------------------------------------------------------

void clTampon::Remplir (clDiskData& DD) {
  memcpy(Adresse, DD.Contenu(), Taille);
};

//---------------------------------------------------------------------//
//                      CLASSE    clEquipment                          //
//---------------------------------------------------------------------//
class clEquipment {

protected:
struct  equipment {
      unsigned int Floppy    : 1;      unsigned int Coproc    : 1;
      unsigned int PointDev  : 1;      unsigned int reserved1 : 1;
      unsigned int VideoO    : 2;      unsigned int FloppyNb  : 2;
      unsigned int reserved2 : 1;      unsigned int SerialNb  : 3;
      unsigned int JoyStick  : 1;      unsigned int reserved3 : 1;
      unsigned int ParalNb   : 2;
} far * Equip_Flag;
byte far * HDnbre;

public:
  clEquipment (void) {
```



```
      // ajustage des pointeurs sur la zone de données du BIOS
      Equip_Flag = (struct equipment far *) 0x00400010;
      HDnbre = (byte far *) 0x00400075;
  }
  // nombre de floppies: 0 -> 1 disquette, 1 -> 2 disquettes
  byte FloppyNbre (void) { return 1 + Equip_Flag->FloppyNb; }
  byte HardDNbre (void) { return * HDnbre; }
  byte DosToBios(byte Lettre);
} ;

//---------------------------------------------------------------------
//                fonctions associées à clEquipment
//---------------------------------------------------------------------

byte clEquipment::DosToBios(byte Lettre) {
// DOS appelle les disques A, B, C, ou 0, 1, 2, 3....
// BIOS les appelle 0, 1, ..., 80, 81.... (en hexa)
// cette fonction tente d'assurer la conversion de notation selon le contexte
// exemple avec 2 disques durs, on peut avoir 80 -> C: et D:, 81 -> E:
// ou bien 80 -> C: et E:, 81 -> D:, selon le statut des partitions
  Lettre = toupper(Lettre);
  if  (Lettre == ''A'')                          return 0;
  if ((Lettre == ''B'') && Equip_Flag->FloppyNb) return 1;
  if ((Lettre == ''C'') && HardDNbre())          return 0x80;
  if  (Lettre == ''D'')
            switch (HardDNbre()) {
                  case 1 : return 0x80;
                  case 2 : return 0x81;
            }
  // défaut
  return -1;
}

//---------------------------------------------------------------------
void Note (int Hauteur, int Duree, int Silence) {
  sound(Hauteur); delay(Duree); nosound(); delay(Silence);
}
void Tutut (void) { Note(2500,50,50); Note(1500,50,50); Note(2500,50,50); }
//---------------------------------------------------------------------

void main (int argc, char ** argv) {
  bool Is_Virus = FALSE;
  byte Disque = 0;
  clEquipment Equip;

// test de l'objet Equip
//   cout << endl << «Ressources de la machine»
//     << endl << Equip.FloppyNbre() << « floppy(s)»
//     << endl << Equip.HardDNbre()  << « hard-disk(s)» << endl;

  cout << endl << «Identification et traitement de VIRUS DE PARTITION»
  << endl << «Avril 1991, J-L Nail et la Revue de l'Utilisateur de l'IBM-PC»;

  // analyse de l'argumentaire de la ligne de commande
  // -------------------------------------------------
  if ((argc == 2) &&
        strchr(«ABCD», toupper(argv[1][0])) &&
        (argv[1][1] == '':'')) {              // c'est bon!
           Disque = Equip.DosToBios(argv[1][0]);
  } else {
      cout << endl << «Usage: RELIVE D:»;
      exit(1);
```

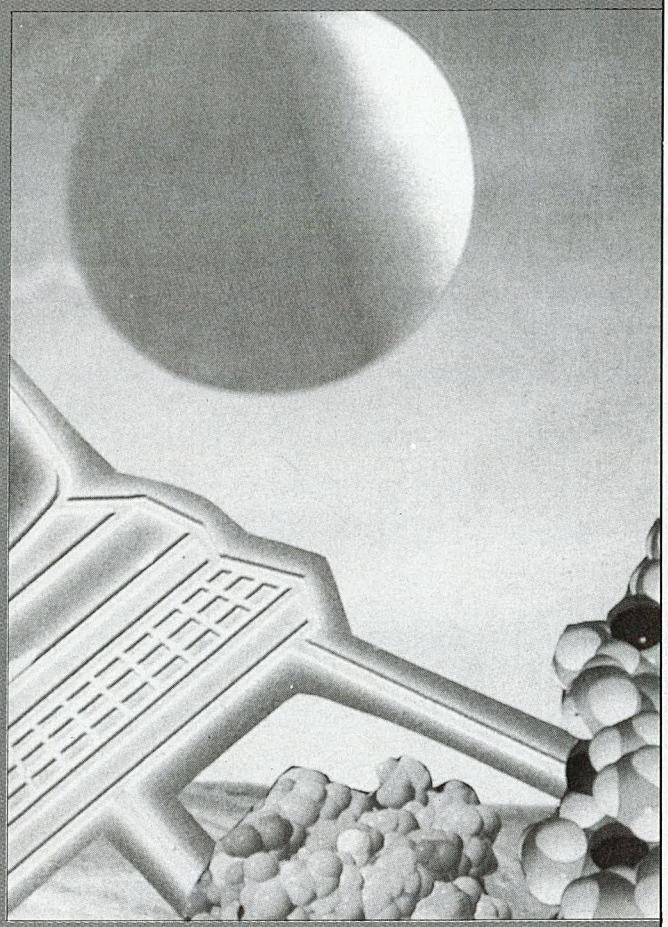
```
}

cout << endl << «Traitement du disque n° « << hex << Disque << dec;

// déclaration de l'objet secteur de boot de disque à tester
clVirusBoot VirusBoot (Disque);

// on commence par OHIO
// VirusBoot.VirusID (OHIO_ID, strlen(OHIO_ID), OHIO_OF);
// écriture synonyme
VirusBoot (OHIO_ID, strlen(OHIO_ID), OHIO_OF);

cout << endl << endl << «Lecture du secteur 0 absolu»;
if (VirusBoot.Lire()) {
    cout << endl << «Erreur de lecture»;
    exit(1);
}
VirusBoot.Bavard();
VirusBoot.Afficher();

    //-------------------------//
    //    DETECTION DE OHIO    //
    //-------------------------//

if (Is_Virus = VirusBoot.IsVirus())  {
    Tutut();
    cout << endl << «Le virus OHIO a été détecté»;
    // test avec seconde signature
    VirusBoot (JMP_OHIO, 2, 0);
    if (!VirusBoot.IsVirus()) {
        cout << « mais il a été traité»;
        Is_Virus = FALSE;
    }
} else cout << endl << «Pas de OHIO»;

if (Is_Virus) {

    //---- ---------------------//
    //    TRAITEMENT DE OHIO    //
    //--------------------------//

    cout << endl << endl << «Lecture de la piste du virus OHIO»;

    // création de l'objet local VraiBoot, c'est la piste 41 du disque
    // on va lire volontairement les 6 secteurs, un seul aurait suffi
    clDiskData VraiBoot(40, 33, 6, Disque, 0);
    VraiBoot.Lire();
    VraiBoot.Afficher();

    // création d'un fichier de capture, local au bloc d'instructions
    ofstream CaptureFile;
    // ouverture en mode binaire, création ou écrasement
    CaptureFile.open(«Capture.bin», ios::binary | ios::out | ios::trunc);
//  VraiBoot.Buffer.Write(CaptureFile);
    CaptureFile << VraiBoot.Buffer;
     CaptureFile.close();

    // création d'un objet tampon, local au bloc d'instructions
    // tampon mémoire intermédiaire
    clTampon Tampon (VraiBoot.Taille() / SECTOR);
```



```
    // récupération du vrai secteur de boot en vu de sa retouche
    VraiBoot >> Tampon ;

    // correction de la signature DOS dans le tampon
//  Tampon.Contenu() [0x1fe] = DOS_ID;
//  Tampon.Contenu() [0x1ff] = DOS_ID >> 8;
    // écriture synonyme
    Tampon [0x1fe] = DOS_ID;
    Tampon [0x1ff] = DOS_ID >> 8;

    // injection du tampon dans l'objet VirusBoot
    // seul le 1er des 6 secteurs est transféré, sans débordement
    VirusBoot << Tampon;

    // affichage pour le contrôle
    VraiBoot.Muet();
    VirusBoot.Afficher();

    // on pourra gérer les erreurs d'écriture
    VirusBoot.Ecrire();
    // sortie
    exit(0);
}

    //-------------------------//
    // détection de EDV / Cursy //
    //-------------------- ------//

if (!Is_Virus) {
    VirusBoot (EDV_ID, strlen(EDV_ID), EDV_OF);
    if (VirusBoot.IsVirus())  {
        Is_Virus = TRUE;
        Tutut();
        cout << endl << «Le virus Cursy / EDV a     détecté»;
    } else cout << endl << «Pas de Cursy / EDV»;
}
if (Is_Virus) {

    //-------------------------//
    // traitement de EDV / Cursy //
    //-------------------------//

    clDiskData VraiBoot(39, 8, 1, Disque, 1);
    VraiBoot.Bavard();
    VraiBoot.Lire();
    VraiBoot.Afficher();
    if (VraiBoot.IsDos()) {
        cout << endl << «Récupération du secteur 0»;
        VraiBoot >> VirusBoot;
        // on pourra gérer les erreurs d'écriture
        VirusBoot.Ecrire();
    } else cout << endl << «Secteur 0 non récupérable»;
    exit(0);
}

    //-------------------------//
    //   détection de STONED   //
    //-------------------------//

if (!Is_Virus) {
```

```
VirusBoot (STONED_ID, strlen(STONED_ID), STONED_OF);
if (VirusBoot.IsVirus()) {
        Is_Virus = TRUE;
        Tutut();
        cout << endl << «Le virus STONED a été détecté»;
    } else cout << endl << «Pas de STONED»;
}
if (Is_Virus) {

    //-------------------------------//
    //  traitement de STONED  //
    //-------------------------------//

    clDiskData VraiBoot(0,                  // piste 0
                Disque < 0x80 ? 3 : 7,    // si floppy secteur 3
                                          // sinon disque dur secteur 7
                1, Disque,                // taille 1 secteur
                Disque < 0x80 ? 1 : 0);   // si floppy tete 1
                                          // sinon disque dur 0

    VraiBoot.Lire();
    VraiBoot.Bavard();
    VraiBoot.Afficher();
    if (VraiBoot.IsDos()) {
        cout << endl << «Récupération du secteur 0»;
        VraiBoot >> VirusBoot;
        VirusBoot.Ecrire();
        // on pourra gérer les erreurs d'écriture
    } else cout << endl << «Secteur 0 non récupérable»;
    exit(0);

}
```

**RELIVE.CPP Page 13**




## COUIC

### L'outil à désassembler

Que vous **chassiez le virus** ou que vous cherchiez à comprendre comment fonctionne un programme **COM** ou **EXE**, que vous soyez programmeur professionnel ou débutant, il vous faut un bon **désassembleur immédiat**, véritable **fenêtre ouverte** sur le code machine.

— Vous devez pouvoir **désassembler un programme** en ROM, en RAM sur disque ou sur disquette en spécifiant une adresse, **un numéro d'interruption**, un nom de fichier ou un **secteur** et faire défiler le listing de désassemblage ou de dumping en avant et en arrière.

— Vous devez disposer d'instructions **puissantes** de localisation automatique d'**interruption**, de **suivi de jump**, ou de **recherches** de chaînes et de données.

— Vous devez disposer de commandes suffisamment simples pour être apprises et manipulées sans difficulté mais assez efficaces pour faciliter la lecture du code et effectuer des **marquages** dans les zones de données.

— Vous devez pouvoir générer des **fichiers séquentiels** utilisés par les désassembleurs symboliques.

— Vous devez disposer du **code source commenté** de votre désassembleur de façon à l'adapter à vos besoins spécifiques.

**COUIC-DESAS permet tout cela** et plus encore grâce à ses fonctions spéciales et ses tables descriptives d'interruptions.

**Exceptionnellement COUIC est vendu avec ses sources en langage "C"** : plus de **9 000 lignes** de code réutilisable dans vos applications et sa documentation pour seulement **340 F (TTC)**.

**COUPON**
**à adresser à La Revue de l'Utilisateur de l'IBM PC**
**2, rue d'Amsterdam**
**75009 Paris**
**Tél. : 47.64.07.57. - 42.67.93.80.**

**Nom**.....

**Adresse**

.....

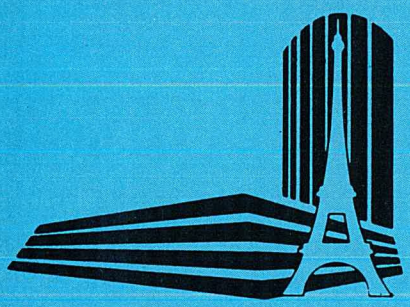
**Code Postal**..... **Ville**

**Téléphone**.....

**Veuillez m'envoyer** ..... **logiciel(s) COUIC au prix unitaire de 340 F. (TTC).**

**Ci-joint un chèque de** ..... **F. En règlement à Softin Communications.**

# 11e Congrès de l'E.A.O. et du Multimédia



## Les 7, 8 et 9 Janvier 1992
## Palais des Congrès
## Porte Maillot Paris 17e

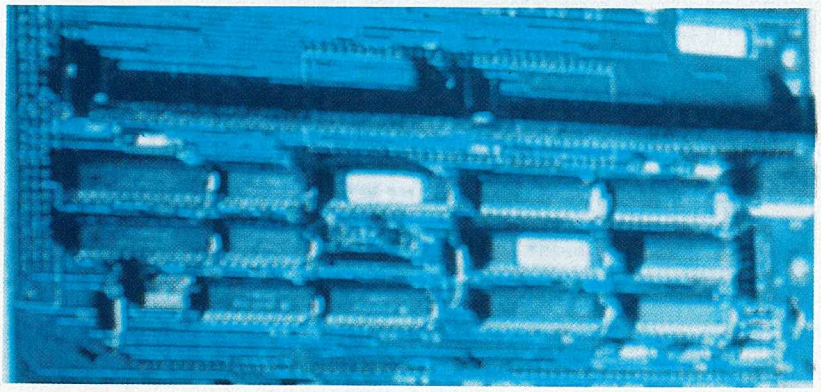
**POUR RETENIR UN STAND**

**Le Journal de la Formation Continue et de l'E.A.O.**
**2 rue d'Amsterdam**
**75009 Paris**
**Tél. : 47.64.07.57. - 42.67.93.80.**

# XMS 2.0

**Ou les secrets de HIMEM.SYS de MICROSOFT. XMS étant le sigle de eXtended Memory Specification, autrement dit, spécification portant sur la mémoire étendue.**

DANIEL VASSEUR



La seule vue du X suffit donc à exclure toutes les machines de type PC du champ d'application de cette norme d'utilisation : on ne le dira jamais assez, les microprocesseurs 8086 ne savent adresser que de la mémoire conventionnelle, par définition limitée à 1 MO.

D'autre part, si le terme « norme » surprend, sachez qu'il est justifié, car quand on s'appelle MICROSOFT et que l'on verse dans le domaine public une série de préconisations ou plus exactement une méthode de travail déjà largement employée en interne, il y a de fortes chances pour que le document de référence se transforme très rapidement en livre de chevet pour de nombreux programmeurs système.

A la date d'aujourd'hui, il est quasiment impossible d'éviter XMS irrémédiablement entraînée dans le tourbillon commercial de WINDOWS, mais aussi avec des produits plus spécifiques comme le linker RTLink Plus de Pocket Soft Inc ou les versions récentes des drivers NOVELL.

## Que fait XMS ?

XMS essaye d'harmoniser l'utilisation des ressources mémoire de la machine que DOS ne sait pas prendre en compte. XMS peut se résumer en 3 points principaux :

1) une méthode d'accès à la mémoire étendue des machines ;

2) une extension de l'espace de travail alloué à DOS ;

3) une normalisation de l'utilisation dynamique de l'espace au-dessus de la mémoire vidéo ;

le tout étant uniquement destiné à l'environnement DOS.

Bizarrement, XMS tente de régler certains problèmes rencontrés sous DOS sur des machines qui n'en sont pas à l'origine. Le paradoxe se situe au niveau de la formidable armada de machines 8088 que l'on ne peut ignorer et qui pourtant n'est pas concernée.

Du fait de la contrainte d'hérédité, peu de programmes acceptent de tirer réellement profit d'XMS. Il faut dire aussi que dans ses implémentations, elle n'apporte que peu de choses nouvelles au programmeur si ce n'est une garantie de fonctionnement des programmes dans un ensemble cohérent, pour autant que tout le monde respecte la règle du jeu, ce qui n'est pas encore une réalité quotidienne.

Pour simple exemple, prenons l'accès à la mémoire étendue, qui se fait désormais à travers XMS chez MICROSOFT et bien d'autres encore, alors que le géant IBM persiste dans l'utilisation d'un système concurrent et totalement incompatible : le fameux ''VDISK Allocation Scheme''.

Comme il est impossible de faire tourner en l'état un programme DOS directement en mémoire étendue - l'espace de 15 MO qui se trouve au-delà de la mémoire conventionnelle de nos machines ne peut donc servir que comme espace de stockage d'informations inertes - XMS trouve là un des obstacles les plus importants à sa reconnaissance et donc à sa diffusion.

De plus, en l'absence d'une stratégie commune aux 2 géants de la micro-informatique, d'autres éditeurs de logiciels comme PHAR LAPS se sont depuis longtemps emparés du gâteau mémoire étendue qui ont favorisé par exemple l'éclosion de la norme VCPI (Virtual Control Process Interface) dont LOTUS 3 tire grandement profit.

Pourtant XMS offre 2 autres fonctionnalités intéressantes que l'on pourrait regrouper sous le vocable commun de ''extension de la zone de travail du DOS''.

Pensez quand même que si cette nouvelle possibilité était vraiment magique, nous le saurions depuis longtemps, et vous aussi par contrecoup.

Nous ne tarderons pas à lever le voile sur ce mystère, mais auparavant, regardons comment se présente XMS.

## Le driver

Comme il s'agit d'une fonction non naturelle du DOS, la couche XMS doit être chargée en mémoire par un ordre DEVICE dans le fichier CONFIG.SYS.

Ainsi installé au chœur du système d'exploitation, XMS devient alors accessible comme tout autre ressource classique du DOS.

Tout comme le célèbre Monsieur Jourdain, nombreux sont les utilisateurs qui mettent XMS à contribution en ignorant même jusqu'à son existence : le plus souvent, l'ordre de chargement du driver a été soumoisement placé dans le fichier CONFIG.SYS lors de la procédure d'installation de WINDOWS 3.

Chez MICROSOFT, le driver s'appelle HIMEM.SYS, mais il existe d'autres implémentations comme celle de QUARTERDECK avec QEXT.SYST.

Le driver XMS a comme principale mission de gérer les accès à la mémoire étendue en régissant des EMB (Extented Memory Block).

En ce domaine, la seule innovation marquante porte sur la présence d'un protocole d'accès qui garantit un usage non conflictuel de la mémoire étendue.

Par opposition avec ce qui se passait au temps glorieux de l'interruption 15h du BIOS de nos AT (interruption toujours mise à contribution par le VDISK d'IBM), un programme confiant ses allocations et libérations de zones au driver XMS sera sûr de ne pas piétiner sauvagement les plates-bandes de ses

voisins.

La programmation s'en trouvera d'ailleurs simplifiée, puisque le programmeur manipulera désormais des "handles" un peu à la manière de l'accès aux fichiers du DOS.

La seconde mission du pilote XMS consiste à étendre d'environ 64 KO l'espace adressable par le microprocesseur, et de mettre ce polder à disposition des programmeurs. Cet espace de travail supplémentaire pour DOS est désigné par le vocable de HMA (High Memory Area).

Le HMA dispose d'un emplacement fixe : le premier segment de la mémoire étendue. En effet, par une bizarrerie de fonctionnement du système d'adressage, il est possible d'accéder cette zone de 64 KO sans faire basculer le microprocesseur en mode protégé.

Au niveau matériel, cela se passe par la prise en gestion de la ligne A20 qui autorise l'adressage au-delà du MO, de sorte que le segment débutant en FFFF:0010 soit accessible comme tout segment mémoire classiquement implanté. La taille utilisable du HMA se trouve donc égale à 65536 – 16 = 65520 octets.

Malheureusement l'accès au HMA n'est possible que pour un seul programme à la fois. MICROSOFT recommande donc de prévoir une option du driver (/HMAMIN) qui permette d'optimiser l'utilisation du HMA en refusant son accès à tout programme de taille insuffisante.

Troisième et dernier apport d'XMS : les Upper Memory Block, autrement dit les UMB qui correspondent aux espaces non utilisés dans l'intervalle [640 – 1024] KO.

Dans cette plage adressable cette fois par les microprocesseurs INTEL en mode réel sans gestion spécifique d'un dispositif hardware donné, un certain nombre d'emplacements sont déjà occupés.

C'est ainsi que l'on trouve la ROM BIOS en F000:0h ou encore la mémoire vidéo d'une carte CGA en B800:0h, sans compter les différentes cartes additionnelles comme les contrôleurs ESDI et autres adaptateurs réseaux. Dans cette jungle se trouvent des clairières : les UMB.

Malheureusement, les UMB ne sont disponibles qu'avec certaines implémentations d'XMS, et de toutes façons, uniquement sur machines 80386, là où il est possible d'activer un mapping mémoire qui fera croire à la disponibilité logique de mémoire dans des emplacements qui n'en contiennent pas physiquement.

Et oui, la mise à disposition des UMB nécessite une programmation qui dépasse de très loin la simple implémentation d'un driver XMS de base, driver d'abord prévu pour fonctionner sur toutes les machines concernées, donc aussi bien les 286 que les 386.

Les UMB, lorsqu'ils sont implémentés, sont alors accessibles par tout programme en faisant la demande au driver XMS. A ce niveau, il y a une réelle extension des capacités traditionnelles du DOS que l'on croyait depuis toujours limité à 640 KO car, à la différence avec ce qui se passe en matière d'EMB, les UMB peuvent très bien servir d'abri au code des programmes TSR par exemple.

Il s'agit en réalité pour XMS d'entériner ce que des éditeurs de logiciels de moindre importance avait mis au point depuis déjà un certain temps, à savoir les gestionnaires de mémoire étendue que nous proposent entre autres QUALITAS, ALL Computers ou encore QUATERDECK.


# LES DISQUETTES DE LA REVUE
## GESDOC

**Logiciel de recherche documentaire (voir page .....).**
**Livré sur disquette 3.5'' programme et sources utilisable directement ou modifiable par vous-même.**
**Permet la recherche par auteur, sujet, titre, éditeur, mots-clés,...**
**Remplace des logiciels qui sont vendus dix fois plus cher...**
**Une application générée à l'aide de ATELIER !**

---

**COUPON**
**à adresser à La Revue de l'Utilisateur de l'IBM PC**
**2, rue d'Amsterdam**
**75009 Paris**
**Tél. : 47.64.07.57. - 42.67.93.80.**

**Nom** ..... **Prénom** .....
**Adresse** .....
.....
**Code Postal** ..... **Ville** .....
**Téléphone** .....
**Veuillez m'envoyer ..... logiciel(s) GESDOC au prix unitaire de 450 F (TTC).**
**Ci-joint un chèque de ..... F. En règlement à Softin Communications.**

En ces temps de pénurie mémoire, tout est bon pour accroître les possibilités de travail du DOS, et il suffit de juger sur le succès des produits des trois éditeurs précités pour admettre qu'il devenait urgent de mettre un peu d'ordre dans l'organisation de la fête.

La reprise en main commence donc par une interface de programmation qui a le mérite d'être préconisée par MICROSOFT et donc d'être reconnue tôt ou tard par tout un chacun (puissent le tout et le chacun s'y soumettre le plus tôt possible pour le plus grand bien des utilisateurs et programmeurs !).

## Comment parler au conducteur

L'utilisation d'XMS se fait à travers un protocole qualifié d'API pour Application Programming Interface.

Tous les appels aux services XMS sont groupés derrière un point d'entrée unique qu'il faudra bien sûr identifier préalablement à toute requête. Une fois cette étape franchie, il suffit d'appeler la routine maître XMS en lui passant des paramètres par registres du microprocesseur interposés.

La détection du pilote XMS se fait par la simple séquence de code listée en figure n° 1.

FIGURE N° 1
**Détection du driver XMS**

```
Detect_XMS:     mov     ax, 4300h
                int     2Fh
                cmp     al, 80h
                jne     NoXMSDriver
```

Vous y remarquerez l'appel à l'interruption MULTIPLEX (2Fh) déjà utilisée par MICROSOFT dans la gestion des files d'attente de la commande PRINT.

Une fois l'existence du pilote révélée, il faut obtenir l'adresse du point d'entrée de la routine principale des services XMS. Là encore il faut faire appel à une séquence de code immuable, celle que nous avons reproduite en figure n° 2.

Certains drivers XMS laissent la ligne A20 active en permanence, de sorte qu'il soit possible de visualiser le contenu du segment des EMB (Extended Memory Block).

Dès lors, les 18 services XMS (voir figure n° 3) sont accessibles par simple appel à la routine placée à l'adresse AdresseXMS que le programme vient d'obtenir.

Les services XMS concernent bien sûr les 3 fonctionnalités principales de la norme (distribution d'HMA, EMB et UMB) mais fournissent également une interface de programmation pour la prise de contrôle de la ligne A20.

Sur un 286 travaillant en mode réel, seule une partie des possibilités d'adressage est validée par souci de compatibilité avec le 8088. C'est ainsi que 20 lignes d'adresses sur 24 sont utilisées pour générer une adresse mémoire physique : les lignes A0 à A19. Dans ce cas, il y a 1 048 576 cases mémoire d'1 octet accessibles, ce qui correspond bien à 1 MO.

Si l'on demande la génération d'une adresse supérieure à cette limite, il y a débordement et remise à zéro du compteur d'adresses, de sorte que la valeur FFFF:0010 se voit transformée en 0000:0000. On se retrouve alors au point de départ de la mémoire adressable par le microprocesseur, autrement dit le début de la table des vecteurs d'interruptions.

FIGURE N° 2
**Adresse de la routine XMS**

```
AdresseXMS      dd      ?

OuEstXMS:       mov     ax, 4310h
                int     2Fh
                mov     word ptr [AdresseXMS], bx
                mov     word ptr [AdresseXMS + 2], es
```

FIGURE N° 3
**Les services XMS**

| | | |
|---|---|---|
| 0h | Numéro de version XMS | |
| 1h | HMA : | Allocation |
| 2h | | Libération |
| 3h | Ligne A20 : | Validation globale |
| 4h | | Libération globale |
| 5h | | Validation locale |
| 6h | | Libération locale |
| 7h | | Etat |
| 8h | EMB : | Taille du plus grand |
| 9h | | Allocation |
| 0Ah | | Libération |
| 0Bh | | Recopie |
| 0Ch | | Verrouillage |
| 0Dh | | Déverrouillage |
| 0Eh | | Informations diverses |
| 0Fh | | Modification de taille |
| 10h | UMB : | Allocation |
| 11h | | Libération |

Pour accéder au code placé dans le HMA, il faut pouvoir valider la génération d'adresses supérieures à F000:FFFF. Ce résultat s'obtient en validant la ligne d'adresse A20, de sorte que la case mémoire placée en FFFF:FFFF devienne la dernière accessible.

En pratiquant de la sorte, le microprocesseur placé en mode réel est alors capable d'accéder à (65536 * 16) + (65536 – 16) = 1 114 096 octets. Certains drivers XMS laissent la ligne A20 active en permanence, de sorte qu'il soit possible de visualiser le contenu du segment HMA sans écrire le programme compliqué.

QEXT.SYS fonctionne ainsi, et pour vérifier que la ligne A20 est validée, il suffit de lancer DEBUG et de demander l'affichage de la mémoire situé en FFFF:10 puis de comparer avec les valeurs contenues en 0:0.

Nous avons reproduit en figure n° 4 les affichages obtenus de la sorte, qui font ressortir la signature de notre RAMDRIVE.SYS (celui qui s'appuie sur XMS et qui est fourni avec WINDOWS 3) par opposition au contenu de la table des vecteurs d'interruption placée en 0:0.

## L'utilitaire du mois

L'utilitaire du mois n'est pas vraiment un utilitaire bien que d'une utilité certaine, au moins pédagogique.

Ecrit en C, il affiche un certain nombre d'indications sur la présence et l'implémentation du driver XMS.

Il montre comment à l'aide d'un langage de haut niveau accéder aux ressources qu'XMS nous proposent et la lecture des sources vous prouvera que l'interfaçage n'est pas spécialement difficile.

Pour éviter la mise au point d'une fonction en assembleur, nous avons rédigé la fonction Appel_XMS à l'aide des spécificités des produits BORLAND.

Les propriétaires de compilateurs C qui ne supportent pas les accès directs aux registres du microprocesseur nous excuseront : ils développeront par eux-mêmes les équivalents assembleurs.

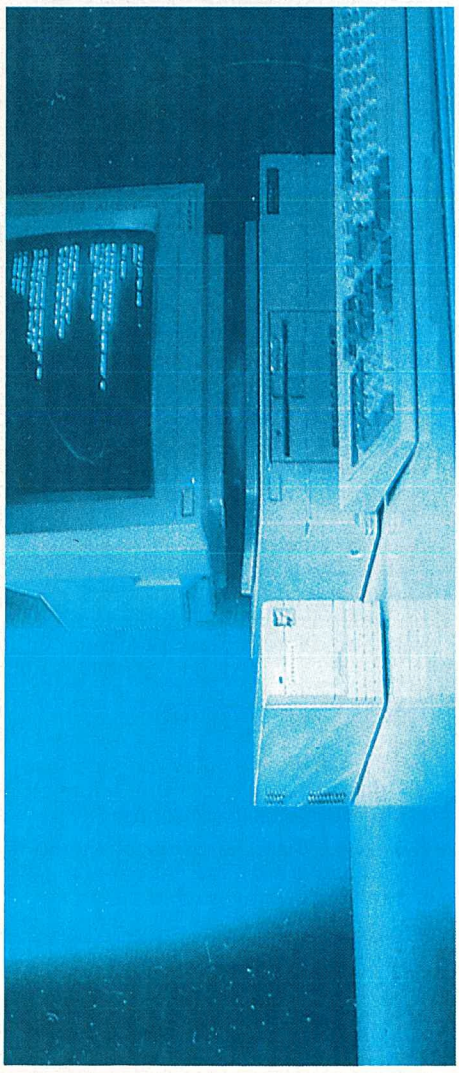
Les procédures d'échange mémoire conventionnelle étendue se développent ensuite rapidement à l'aide d'un couple de fonctions complémentaires assurant l'importation et l'exportation des données.

La procédure de rédaction est voisine de celle déjà présentée en Mars 1990 dans ces mêmes colonnes (''Mémoire étendue sur PC AT''), à ceci près que la présence d'XMS nous dispense de générer des adresses sur 24 bits : les codifications portent désormais sur un mot (16 bits) servant de ''handle'' complété d'un offset sur... 32 bits.

N'oublions pas que nous sommes désormais susceptibles de travailler sur la mémoire de machines 386 dont les microprocesseurs sont capables d'adresser des espaces gigantesques (du grec GIGA !).

FIGURE N° 4
**Visualiser le HMA**

```
-D FFFF:0
FFFF:0000  EA 38 05 C0 09 30 34 2F-30 39 2F 39 30 00 FC 00   j8.@.04/09/90.|.
FFFF:0010  00 00 00 56 44 49 53 4B-33 2E 33 00 00 00 00 00   ...VDISK3.3.....
FFFF:0020  00 00 00 00 00 00 00 00-08 00 01 00 00 00 40 04   ..............@.
FFFF:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
FFFF:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
FFFF:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
FFFF:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
FFFF:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-D 0:0
0000:0000  05 57 75 02 5C 07 70 00-1B 2C DF 49 5C 07 70 00   .Wu.\.p..,_I\.p.
0000:0010  5C 07 70 00 54 FF 00 F0-A6 EA 00 F0 A6 EA 00 F0   \.p.T..p&j.p&j.p
0000:0020  AB 00 19 0D AD 2B DF 49-1A 02 19 0D C3 1E 7F EF   +...-+_I....C..o
0000:0030  2A 03 19 0D B2 03 19 0D-3A 04 19 0D 5C 07 70 00   *...2...:...\.p.
0000:0040  12 10 7F EF F6 04 7C 26-41 F8 00 F0 CA 0A C0 09   ...ov.|&Ax.pJ.@.
0000:0050  39 E7 00 F0 C7 04 EC E8-2E E8 00 F0 D2 EF 00 F0   9g.pG.lh.h.pRo.p
0000:0060  00 E0 00 F0 A2 00 EC E8-6E FE 00 F0 91 05 1F 10   .`.p".lhn^.p....
0000:0070  53 FF 00 F0 A4 F0 00 F0-22 05 00 00 00 00 00 00   S..p$p.p".......
```



## Humeur

XMS dans sa version 2.0 est datée du 23/08/88, et est disponible auprès du service client de MICROSOFT.

Mais XMS n'est pas le fruit de ce seul concepteur de logiciels : INTEL, LOTUS et AST sont également dans le coup, tout comme dans celui de l'EMS d'ailleurs.

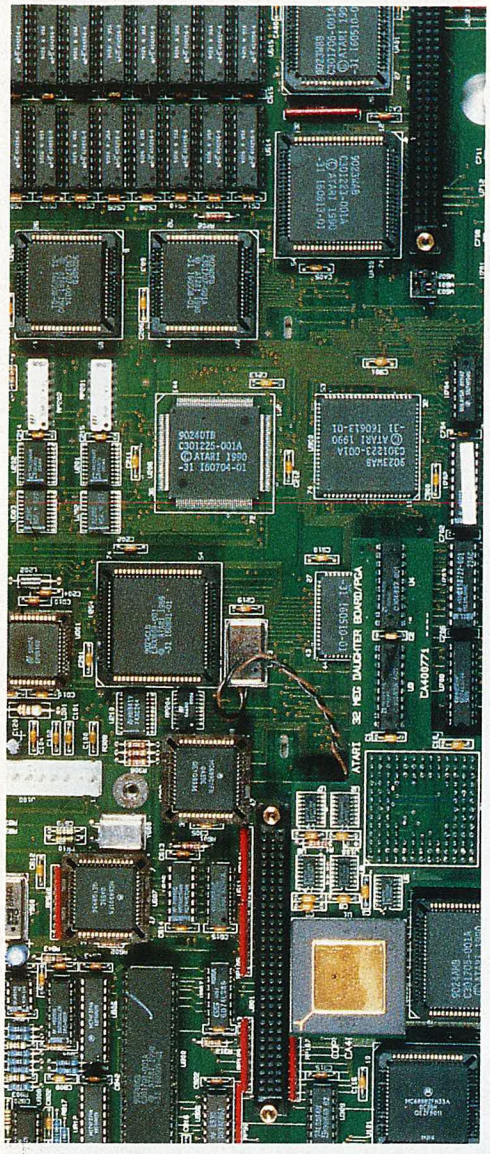
La brochure, épaisse d'une vingtaine de pages, est d'abord destinée aux développeurs et techniciens avec en tout et pour tout 2 ultra courts exemples (nos figures 1 et 2) de programmation... en assembleur.

Elle n'en reste pas moins indispensable, notre magazine n'ayant pas la volonté de reproduire in extenso et in petto le fruit du labeur des autres.

A l'occasion de cet article, nous saluons la volonté récente de MICROSOFT France d'ouvrir enfin la porte du dialogue avec les modestes rédacteurs et développeurs que nous sommes.

Il nous a tout de même fallu 12 mois de persévérence et presque autant d'interlocuteurs différents pour obtenir confirmation de l'existence en France de la brochure traitant d'XMS. Une fois la bonne adresse trouvée (merci encore à BORLAND France de nous l'avoir donnée), nous avons reçu le document tant recherché en moins de 24 heures.

Voici une nouvelle source d'information pour notre magazine : formulons le vœux de ne jamais la voir se tarir.



XMS.C par D. VASSEUR dans La Revue de l'Utilisateur PC

```
#include  <dos.h>
#include  <stdio.h>
#include  <process.h>

typedef   unsigned int   word;
typedef   unsigned char  byte;
typedef   enum {FALSE, TRUE} bool;
typedef   struct {                              /* Sert à la converion BCD -> ASCII */
     word V4:4;                                 /* Attention à l'ordre des champs */
     word V3:4;                                 /* de bits qui peut varier selon */
     word V2:4;                                 /* les compilateurs C */
     word V1:4;
} BCD;

union     REGS CPU;                             /* Pour dialoguer avec le driver */
void far  (*Acces_XMS)(void);                   /* Adresse des services XMS */


void Erreur(byte *Message) {                    /* Affiche un message d'erreur */
     printf(Message);                           /* et met fin au programme */
     exit(1);
}


bool Presence_XMS(void) {                       /* Détecte la présence du driver */
     struct REGPACK R;                          /* XMS et mémorise le point d'entrée */
                                                /* de la routine des services */
     R.r_ax = 0x4300;
     intr(0x2F, &R);
     if (R.r_ax & 0x80) {
          R.r_ax = 0x4310;
          intr(0x2F, &R);
          Acces_XMS = MK_FP(R.r_es, R.r_bx);
          return(TRUE);
     }
     else
          return(FALSE);
}


void Appel_XMS(byte Fonction, word Parm) {      /* Appel des services XMS */
     _AH = Fonction;                            /* Prépare les registres */
     _AL = 0;
     _DX = Parm;
     (*Acces_XMS)();                            /* Passe la main au driver */
     CPU.x.bx = _BX;                            /* Restitue les valeurs reçues */
     CPU.x.ax = _AX;
     CPU.x.dx = _DX;
}


byte *    BCD_ASCII(BCD *V) {                   /* Transforme une valeur BCD en */
     static byte    T[6];                       /* son équivalent ASCII normalisé */
     int            I = 0;                      /* sous la forme xX.XX */

     if (V->V1)                                 /* Evite le 0 initial quand c'est */
```



XMS.C par D. VASSEUR dans La Revue de l'Utilisateur PC

```
          T[I++] = V->V1 + '0';                 /* possible */
     T[I++] = V->V2 + '0';                      /* Octet fort, quartet faible */
     T[I++] = '.';                              /* Ponctuation */
     T[I++] = V->V3 + '0';                      /* Octet faible, quartet fort */
     T[I++] = V->V4 + '0';                      /* Octet faible, quartet faible */
     T[I++] = 0;                                /* Fin de chaîne */
     return(T);                                 /* dont on retourne l'adresse */
}


void main(void) {
     if (! Presence_XMS())
          Erreur("Driver XMS non chargé !\r\n");

     Appel_XMS(0, 0);                           /* Retourne le numéro de version */
                                                /* et de release en notation BCD */
     printf("Driver XMS en version %s ", BCD_ASCII((BCD *)&CPU.x.ax));
     printf("(Révision %s)\r\n", BCD_ASCII((BCD *)&CPU.x.bx));
     printf("Le segment HMA est %s et ", CPU.x.dx ? "présent" : "absent");

     Appel_XMS(0x07, 0);                        /* Etat de la ligne A20 */
     printf("la ligne A20 %sactive.\r\n", CPU.x.ax == 1 ? "" : "in");

     Appel_XMS(0x10, 0xFFFF);                   /* Status des UMB */
     if (CPU.x.dx == 0)
          printf("Aucun UMB disponible.\r\n");
     else
          printf("Il y a %u KO disponibles sous forme d'UMB.\r\n", CPU.x.dx / 64);

     Appel_XMS(8, 0);                           /* Status des EMB */
     if (CPU.x.ax == 0)
          Erreur("Aucun EMB disponible.\r\n");
     else
          printf("Un EMB de %d KO est disponible sur un total de %d KO.\r\n",
               CPU.x.ax, CPU.x.dx);
}
```

# Interfaçage

Les explications toutefois, restent tout à fait valables pour n'importe quel langage produisant des fichiers objets (.OBJ).

Dans un premier temps nous aurons une explication du transfert des données, pour voir ensuite les types de paramètres et enfin nous compléterons par un exemple concret.

## Avertissement

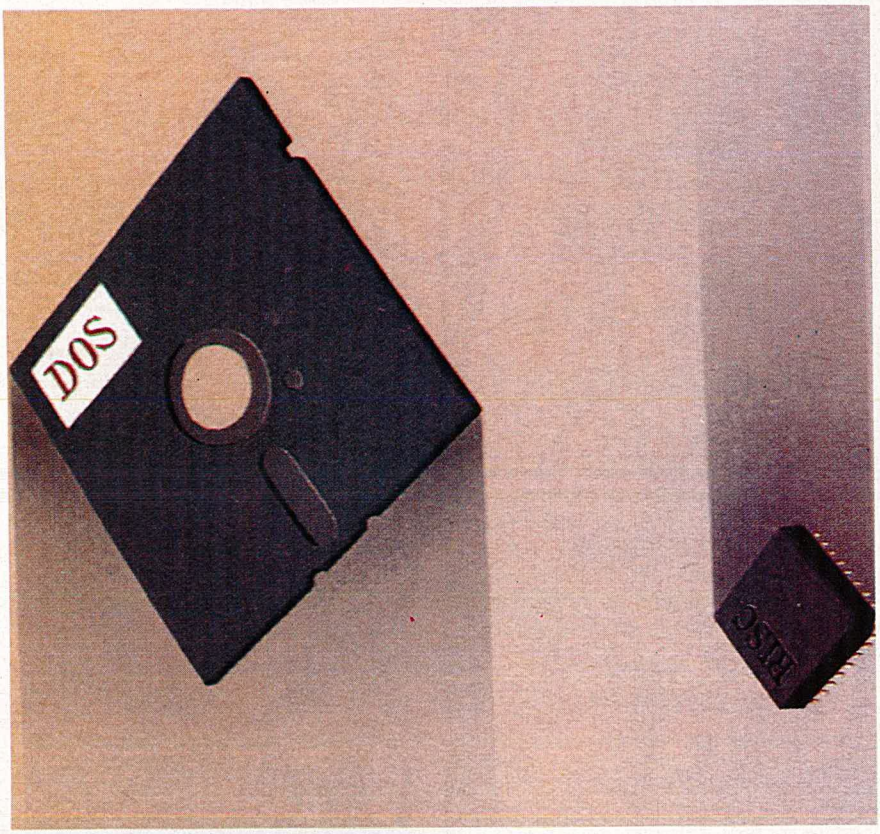
La compréhension de ce texte requiert quelques notions élémentaires en assembleur, ainsi que dans les langages utilisés.

L'application servant d'exemple montrera comment créer un prédicat (sous-programme) en Prolog, C et Assembleur, celui-ci ayant pour but de déterminer la solution d'une équation fort simple dans l'environnement Prolog.

## Le transfert des données

L'utilité des sous-programmes (procédures, fonctions ou prédicats) est de modifier les informations qu'il reçoit et/ou d'en restituer les résultats éventuels.

La méthode que nous emploierons pour échanger ces informations consiste en l'utilisation de la pile (STACK en anglais), via les registres SS (Stack Segment), SP (Stack Pointer) et BP (Base Pointer).




**Le but de cet article est d'expliquer comment s'effectue le transfert de données à des sous-programmes et comment on peut interfacer différents langages.**

**Les langages utilisés sont : TURBO PROLOG V1.1, TURBO C V2.0, TURBO ASSEMBLEUR V1.0, de Borland International.**

JEAN-CHARLES LE TARNEC ET MAGALIE S

Nous allons revoir rapidement le fonctionnement de la pile.

La pile : rappels

Le registre SS des microprocesseurs de la famille APx86 (8086, 80286, 80386,...) délimite un espace mémoire maximum de 64 Ko, mais qui est très souvent de taille inférieure. Ce segment sert de mémoire tampon à court terme pour certains types de variables (variables automatiques en C, variables locales en Pascal, variables non globales en Prolog).

Il sert aussi lors de l'appel du moindre sous-programme. Le seul accès "légal" à cette mémoire est assuré par les instructions assembleur suivantes :

PUSH word
POP word
PUSHF
POPF

(et, sur 80386, PUSHA, POPA, PUSHAD, POPAD, PUSHFD, POPFD).

Ces instructions gèrent cette mémoire en tant que pile LIFO (Last In First Out). Elles n'effectuent que des opérations sur des mots (WORD en anglais) et la pile se détermine des adresses les plus grandes vers les plus petites.

La fin d'un sous-programme doit rendre au pointeur de pile (SP) sa valeur qui précédait l'empilement des paramètres destinés au sous-programme en question dans le cas d'un interfaçage. Cependant, selon les langages, quelques différences apparaissent dans la manière de restaurer la pile.

Le schéma n° 1 récapitule ces quelques notions.

Utilité d'un registre de base : BP

Vous avez sans doute déjà compris comment se fera le passage des données :

— empilement des valeurs nécessaires au sous-programme,

— appel du sous-programme,

— restauration de la pile et retour au programme appelant qui récupère éventuellement certaines variables.

Pour accéder aux données, le sous-programme pourrait utiliser des instructions du genre :

mov ax, [sp]
mov bx, [sp + 2], etc...

Cependant, si le sous-programme possède des variables locales ou s'il utilise lui-même des instructions PUSH ou POP, le registre SP sera modifié de telle sorte que tout adressage par [SP + cte] serait difficile.

(La constante varierait selon l'emplacement relatif des instructions), voir impossible car imprévisible dans le cas d'un sous-programme mal structuré (car l'assembleur le permet !) ou encore dans le cas d'un sous-programme récursif.

La solution consiste à utiliser le registre BP en l'initialisant avec la valeur de SP dès le début du sous-programme. Pour conserver la valeur précédente de BP, dans le cas de programmes récursifs ou à plusieurs "sous-niveaux" plus particulièrement, on effectuera un simple PUSH BP avant l'opération précédente.

Mais il n'y a pas que les paramètres qui transitent sur la pile.

Autopsie d'un CALL

De même que nous avons fait pour les instructions PUSH et POP dans le schéma n° 1, nous pouvons définir les appels et les retours à des sous-programmes (CALL en anglais et en assembleur) comme des pseudos macrocommandes comme le montre le schéma n° 2.

Chaque instruction RET rétablit l'état de la pile telle qu'elle était avant le CALL correspondant, du moment que le corps du sous-programme conserve lui aussi la cohérence au niveau de la pile et que le nombre nb corresponde au nombre d'OCTETS (pas de mots) représentant la taille totale des paramètres passés.


# FORUM IA 91

## les 18 et 19 Juin 1991
## Palais des Congrès
## Porte Maillot Paris 17e

## RESEAUX NEURONAUX : L'ACQUISITION DE CONNAISSANCES

**Méthodologie**

**Le métier d'ingénieur de la connaissance.**
**Méthodes et outils pour l'acquisition de connaissances.**
**Validation des connaissances.**
**Evolution des connaissances.**

**Acquisition automatique de connaissances**

**Méthodes symboliques d'apprentissage automatique.**
**Réseaux neuronaux : apprentissage et extraction de connaissances.**
**Classifieurs génétiques.**

**Applications**

**Réalisations industrielles mettant en évidence :**
**— l'application de méthodologies,**
**— la mise en œuvre d'outils d'aide à l'acquisition de connaissances,**
**— la mise en œuvre d'outils d'acquisition automatique de connaissances.**

**Renseignements, programmes et inscriptions**
**Journal de la Formation Continue et de l'E.A.O.**
**2, rue d'Amsterdam**
**75009 Paris**
**Tél. : 47.64.07.57. - 42.67.93.80.**

L'appel d'un sous-programme écrit dans un autre langage

Les sous-programmes appelés doivent être compatibles avec l'environnement créé par le compilateur du programme appelant. Plusieurs conséquences en découlent. Par exemple, tout sous-programme appelé ayant besoin d'allouer de la mémoire doit le faire par les fonctions standards du langage appelant.

En Prolog

Compilateur :

Pointeurs : longs sur 32 bits (FAR PTR).

Appels : longs sur 32 bits (CALL FAR).

Empilement : à l'endroit.

Restauration de la pile : effectuée par les sous-programmes (RET nb_octets).

Gestion de la mémoire : voir schéma n° 2.

Conséquence pour les sous-programmes :

Assembleur : se conformer aux prescriptions ci-dessus.

Appel (Prolog) : CALL FAR nomproc.

Interface (Assembleur) : PUBLIC nomproc, nomproc PROC FAR.

C :

Configuration : modèle mémoire LARGE.

TC : Options/Compiler/Model/Large.

TCC : -ml.

Le compilateur ne doit pas générer de caractères de soulignement :

TC : Options/Compiler/Code génération/Generate underbars Off.

TCC : -u-.

Attention ! Pour utiliser des fonctions standards des librairies du C, vous devrez préfixer manuellement tout appel à l'une d'elles par un caractère de soulignement "_".

Optimisation des sauts :

TC : Options/Compiler/Optimization/Jump optimization On.

TCC : -O.

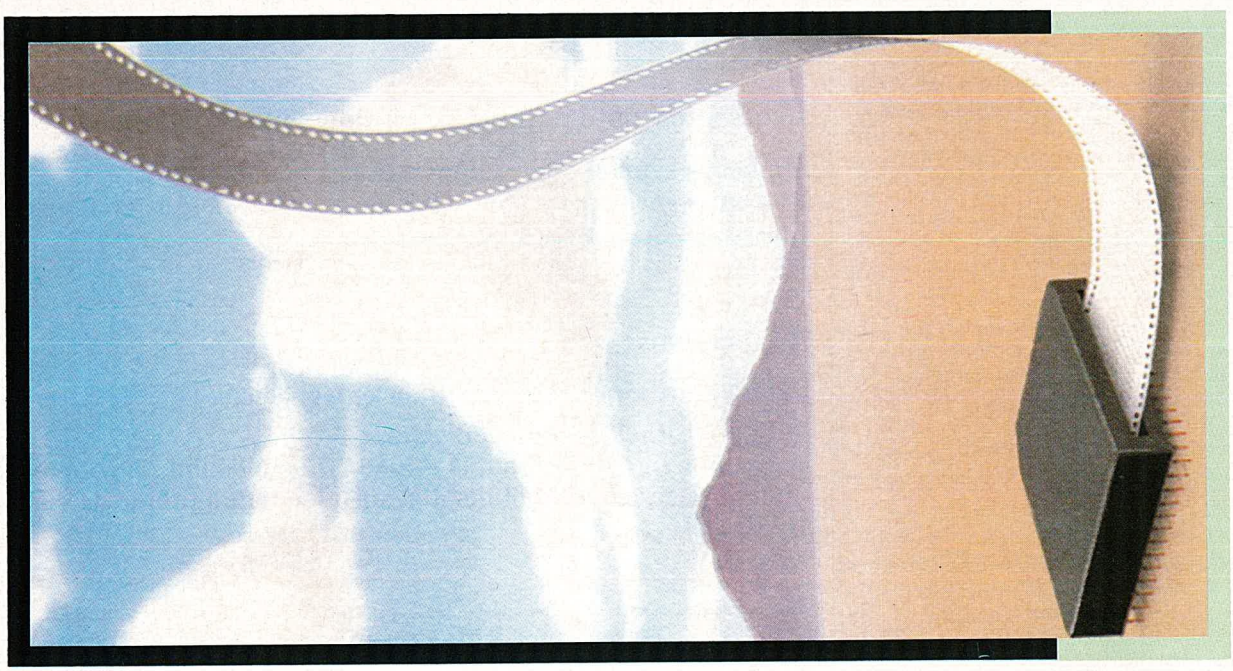
Prolog utilise les registres SI et DI :

TC : Options/Compiler/Optimization/Use register variable Off.

TCC : -r-.

Déclaration pour un prédicat externe en syntaxe BNF.

```
<section-prédicats> ::= global predicates {<global-déf-
prédicat>}*
<global-déf-prédicat> ::= <déf-prédicat>[-{<flow-spec>}*
                                    [language <langage>]]
<déf-prédicat> ::= <nom>[([<liste-noms>])][.]
<flow-spec> ::= (<flow-param-list>)
<flow-param-list> ::= <flow-param> | <flow-param>, <flow-
param-list>
<flow-param> ::= i|o
<langage> ::= asm|c|pascal|fortran
```



Exemple :

global predicates

add(integer, integer, integer) - (i, i, o), (i, o, i), (o, i, i) language asm

Remarques

Les prédicats externes devront alors avoir pour nom add__0, add__1 et add__2.

Il est possible d'appeler ces prédicats avec un autre nom en rajoutant l'expression "as ⟨nom-de-prédicat-externe⟩" avec certaines versions de Prolog.

Le fait de préciser le langage utilisé pour implémenter un prédicat externe permet à Turbo Prolog de gérer convenablement les appels de ceux-ci.

Editions des liens

Fichiers nécessaires :

INIT.OBJ : fichier de base fourni avec Turbo Prolog.

x.OBJ et x.SYM : fichiers créés par Turbo Prolog avec Option/OBJ file.

y.OBJ : fichier ou liste de fichiers des sous-programmes externes.

Syntaxe avec TLINK

TLINK init x y x.SYM, [nom EXE], [fichiersMAP], [autres librairies] prolog.

Les bibliothèques C pouvant être utilisés sont emulib, mathlib et toujours cl.

Tout appel à un prédicat s'effectuera comme suit :

— empilement des paramètres (push...),

— appel du sous-programme (call far nompredicat),

— retour à l'appelant et restauration de la pile (ret nb__octets),

— reprise du programme principal.

Exemple :

Prolog : add (2, 3, Z).

Code assembleur correspondant :

```
mov ax, 0002
mov cx, 0003
push ax
push cx
push ss
lea si, [bp-02]
push si
mov dx, SEGMENT DATA ; de add__0
mov ds, dx
call far add__0
mov si, SEGMENT__DATA ; pour la suite
mov ds, si
```

En C

Compilateur :

Pointeurs : selon le modèle mémoire.

Appels : selon le modèle mémoire.

Empilement : à l'envers.

Restauration de la pile : le code appelant est responsable de la suppression des paramètres dans la pile.

Gestion de la mémoire : selon le modèle mémoire (voir la documentation de TURBO C).

Déclaration pour une fonction externe

En C : ⟨valeur retournée⟩ extern ⟨déclaration standard de fonction⟩

Dans le sous-programme en assembleur : PUBLIC⟨nom⟩

Edition des liens

Fichiers nécessaires : COT, COS, COC, COM, COL ou COH.OBJ selon le modèle mémoire.

x.OBJ : nos fichiers objets séparés par des espaces.

Syntaxe avec TLINK

? : une lettre parmi T, S, C, M, L, H selon le modèle mémoire.

Librairies : FP87 ou EMU, MATHx, Cx,...

TLINK CO? x, [nom EXE], [fichierMAP], [autres librairies] C?

Tout appel à une fonction s'effectue comme suit :

— empilement des paramètres à l'envers (push...),

— appel de la fonction (call [far] nom-fonction),

— retour de la fonction (ret),

— restauration de la pile (add sp, nb__octets).

Exemple : voir schéma n° 3.

Remarques

En TURBO C il est très facile de voir la gestion des appels aux sous-programmes en demandant la traduction du fichier C en fichier assembleur avec la commande :

TCC -S nomfichier. (N'oubliez pas les autres options de la ligne de commandes).

Il est également possible d'utiliser l'assembleur en ligne ou encore d'utiliser le compilateur comme un "super assembleur", mais nous n'utiliserons pas cette technique aujourd'hui.

## Les types des paramètres

Les données transmises à des sous-programmes doivent toujours être d'un type reconnu par ce dernier. Pour vous permettre de les choisir efficacement, nous avons listé les types Prolog et C dans le tableau n° 1.

## Exemple concret

Nous allons maintenant voir un exemple concret de programme utilisant les techniques d'interfaçage. Ce que nous allons faire constitue un exercice de style car nous verrons trois versions du même programme :

— Appel en Prolog / Sous-programme en Prolog.

— Appel en Prolog / Sous-programme en C.

— Appel en Prolog / Sous-programme en Assembleur.

Le problème consiste à résoudre une équation de la forme : X + Y = Z, connaissant deux des trois nombres.

Version Prolog / Prolog

```
/* PROPRO.PRO */
predicates
  add(integer, integer, integer)

clauses
  add(X, Y, Z):-
              free(Z),
              bound(X),
              bound(Y),
              Z = X + Y.
  add(X, Y, Z):-
              free(X),
              bound(Y),
              bound(Z),
              X = Z - Y.
  add(X, Y, Z):-
              free(Y),
              bound(Z),
              bound(X),
              Y = Z - X.

goal
  add(2, 3, Z), write("2 + 3 = ", Z), nl,
  add(2, Y, 5), write("5 - 2 = ", Y), nl,
  add(X, 3, 5), write("5 - 3 = ", X), nl.
```

Manipulations :

— Saisir le texte source Prolog : PROPRO .PRO.

— Créer le fichier.

— Menus : Options/EXE file (auto link) puis Compile.

Version Prolog / C

```
Prolog:
        /* PROC01  .PRO */
        global predicates
          add(integer, integer, integer)-(i, i, o), (i,
o, i), (o, i, i) language c

        goal
          add(2, 3, Z), write("2 + 3 = ", Z), nl,
          add(2, Y, 5), write("5 - 2 = ", Y), nl
          add(X, 3, 5), write("5 - 3 = ", X), nl.

C:
  /* PROC02  .C */
       #pragma inline /* Si on veut passer par
l'assembleur */

    add_0(short int X, short int Y, short int *Z)
     /* Correspond au prédicat add(X, Y, Z) avec X et
Y liés et Z libre */
    {
      *Z = X + Y;
    }
    .
    add_1(short int X, short int *Y, short int Z)
     /* Correspond au prédicat add(X, Y, Z) avec X et
Z liés et Y libre */
    {
      *Y = Z - X;
    }

    add_2(short int *X, short int Y, short int Z)
     /* Correspond au prédicat add(X, Y, Z) avec Y et
Z liés et X libre */
    {
      *X =Z - Y; }
```

Manipulations

Prolog :

— Saisir le texte source Prolog : PROCO1 .PRO.

— Créer les fichiers.

— Menus : Options/OBJ file puis Compile.

C :

— Saisir le texte source C : PROCO2 .C.

— TCC -S -O -ml -r- -u- PROCO2.

— TASM PROCO2.

(On passe par l'assembleur à titre de curiosité).

Edition de liens

Fichiers nécessaires :

— INIT .OBJ : fichier de base de Prolog.

PROCO1.OBJ : créé par Prolog.
PROCO1.SYM : créé par Prolog.
PROCO2.OBJ : créé par C.

Syntaxe avec TLINK :

TLINK INIT PROCO1 PROCO2 PROCO1.SYM, PROCFINI, , PROLOG CL

Version Prolog / Assembleur

Remarque : cette version du programme est un exemple donné par Borland et vous n'avez donc pas à le saisir car il figure sur vos disquettes originales. Nous redonnons quand même les listings (très légèrement modifiés) dans un but de comparaison avec les précédentes versions.

Prolog :

```
/* ADDPRO.PRO */
global predicates
    add(integer, integer, integer)-(i, i, o), (i,
o, i), (o, i, i) language asm

goal
  add(2, 3, Z), write("2 + 3 = ", Z), nl
  add(2, Y, 5), write("5 - 2 = ", Y), nl,
  add(X, 3, 5), write("5 - 3 = ", X), nl.
```

Assembleur:

```
;*************
;*** ADD.ASM
;*************
name add
ADD_TEXT SEGMENT BYTE PUBLIC 'CODE'
     ASSUME     cs:ADD8TEXT
      PUBLIC      ADD_0          ; (i, i, o) mode
d'appel  Z = X + Y
ADD_0 PROC FAR
arg Z:DWORD, Y:WORD, X:WORD = ARGLON1
     push    bp
     mov     bp, sp
     mov     ax, X
     add     ax, Y
     les     bx, Z
     mov     WORD PTR es:[bx], ax
     pop     bp
     ret     ARGLON1
ADD_0 ENDP

      PUBLIC      ADD_1          ; (i, o, i) mode
d'appel  Y = Z - X
ADD_1 PROC FAR
arg Z:WORD, Y:DWORD, X:WORD = ARGLON2
     push    bp
     mov     bp, sp
     mov     ax, Z
     sub     ax, X
     les     bx, Y
     mov     WORD PTR es:[bx], ax
     pop     bp
     ret     ARGLON2
ADD_1 ENDP

      PUBLIC      ADD_2          ; (o, i, i) mode
d'appel  X = Z - Y
ADD_2 PROC FAR
arg Z:WORD, Y:WORD, X:DWORD = ARGLON3
     push    bp
     mov     bp, sp
     mov     ax, Z
     sub     ax, Y
     les     bx, X
     mov     WORD PTR es:[bx], ax
     pop     bp
     ret     ARGLON3
ADD_2 ENDP

ADD_TEXT ENDS
      END
```

Manipulations

Prolog :

— Saisir le texte source Prolog : ADDPRO .PRO.

— Créer les fichiers.

— Menus : Options/OBJ file puis Compile.

Assembleur :

— Saisir le texte source Assembleur : ADD .ASM.

— TASM ADD.

Edition de liens

Fichiers nécessaires :

INIT .OBJ : fichier de base de Prolog.
ADDPRO.OBJ : créé par Prolog.
ADDPRO.SYM : créé par Prolog.
ADD .OBJ : créé par C.

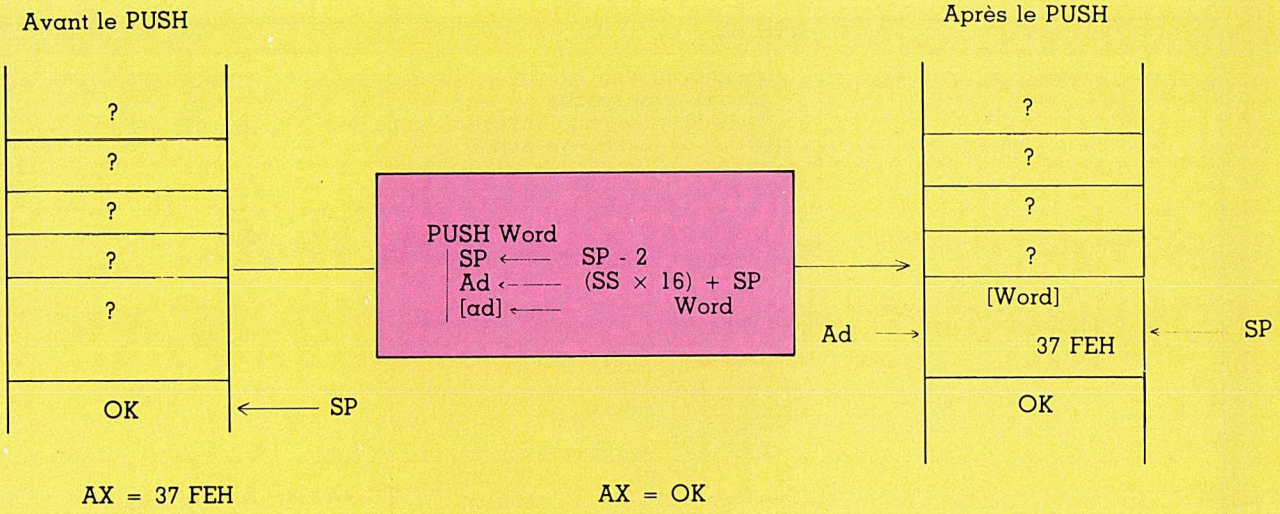
Syntaxe avec TLINK :

TLINK INIT ADDPRO ADD ADDPRO.SYM,ADDFINI,,PROLOG.

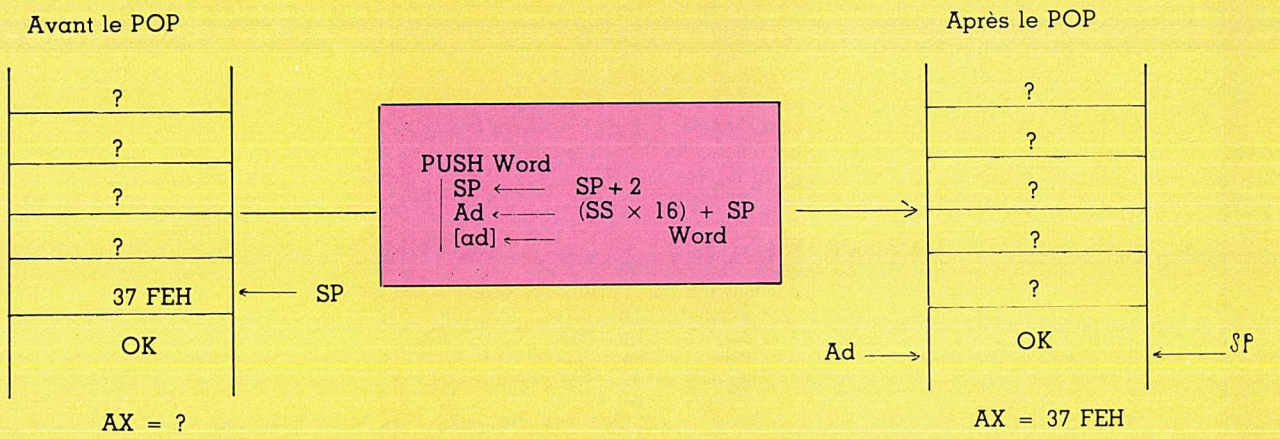
## Conclusion

Nous espérons vous avoir inciter à ce type de programmation et avoir aider les personnes qui se sont senties déroutées par l'absence d'informations explicites dans ce domaine.

SCHEMAS




1a) Exemple d'exécution d'un : PUSH AX avec AX = 37 FEH.




1b) Exemple d'exécution d'un : POP.

? : valeur inconnue ) sans importance.
OK : valeur connue ).
Word : soit un registre 16 bits, soit un mot en mémoire. (Les valeurs directes ne sont supportées que sur le 80386).
Ad : adresse réelle (sur 20 bits).
[x] : contenu de x.

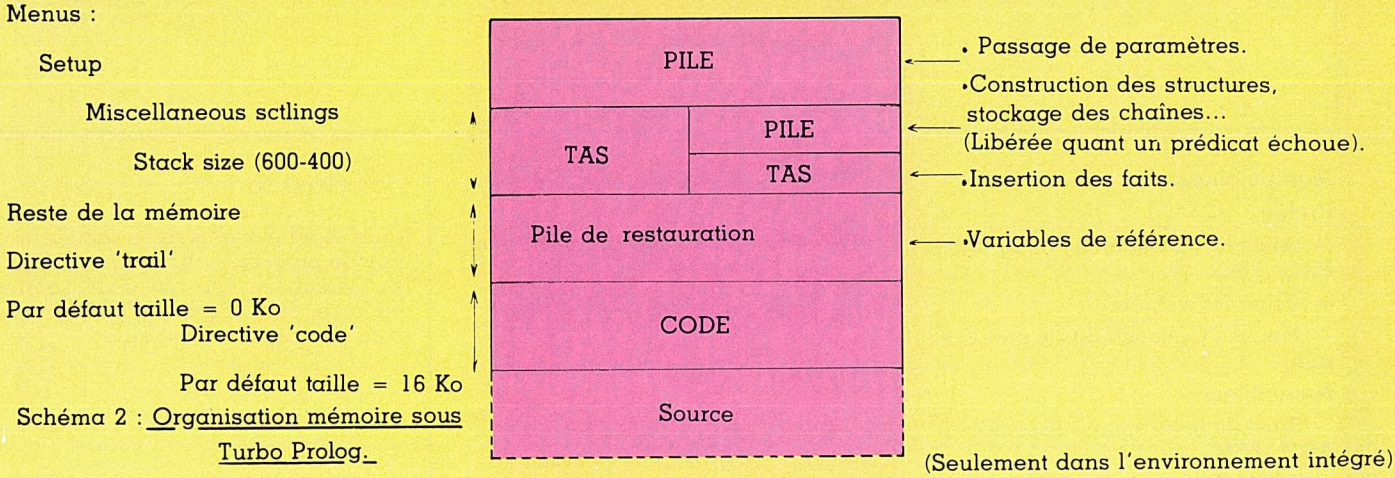



Schéma 2 : Organisation mémoire sous Turbo Prolog.

Schéma 3 : Appel court (SMALL) de la fonction standard "printf" avec les variables registres autorisées et les caractères de soulignement.

| Source C | Equivalent assembleur | Conséquences sur la pile (à l'appel) |
|---|---|---|
| ...<br>int x = 12 ;<br>int y = 14 ;<br>printf ("%i%i,x,y) ; | ...<br>mov si, 12<br>mov di, 14<br>push di<br>push si<br>mov ax, offset - s<br>push ax<br>call near ptr_printf<br>add sp, 6 | SP ⟶ (offset - s )<br>12 (si)<br>14 (di) |

Tableau 1 : Les types des paramètres.

| Types | Taille réelle en octets octets | Taille à l'empilement | | Domaine |
|---|---|---|---|---|
| | | | mots | |
| En Prolog | | | | |
| integer (lié) | 2 | 2 | 1 | - 32 768 + 32 767 |
| integer (libre) | 4 | 4 | 2 | 00000h - FFFFFh |
| real (lié) | 8 | 8 | 4 | (IEEE) |
| real (libre) | 4 | 4 | 2 | 00000h - FFFFFh |
| char (lié) | 1 | 2 | 1 | 0 - 255 |
| char (libre) | 4 | 4 | 2 | 00000h - FFFFFh |
| string (lié) | 4 | 4 | 2 | 00000h - FFFFFh |
| string (libre) | 4 | 4 | 2 | 00000h - FFFFFh (double ptr) |
| symbol (lié) | 4 | 4 | 2 | 00000h - FFFFFh |
| symbol (libre) | 4 | 4 | 2 | 00000h - FFFFFh (double ptr) |
| compound (lié) | 4 | 4 | 2 | 00000h - FFFFFh |
| compound (libre) | 4 | 4 | 2 | 00000h - FFFFFh (double ptr) |
| En C | | | | |
| unsigned char | 1 | 2 | 1 | 0 - 255 |
| char | 1 | 2 | 1 | - 128 - 127 |
| enum | 2 | 2 | 1 | - 32 768 - 32 767 |
| unsigned short | 2 | 2 | 1 | 0 - 65 535 |
| short | 2 | 2 | 1 | - 32 768 - 32 767 |
| unsigned int | 2 | 2 | 1 | 0 - 65 535 |
| int | 2 | 2 | 1 | - 32 768 - 32 767 |
| unsigned long | 4 | 4 | 2 | - 0 - 4 294 967 295 |
| long | 4 | 4 | 2 | - 2 147 483 648 - 2 147 483 64 |
| float | 4 | 4 | 2 | 3 4E - 38 - 3 4E + 38 |
| double | 8 | 8 | 4 | 1 7E - 308 - 1 7E + 308 |
| long double | 10 | 10 | 5 | 3 4E - 4 932 - 1 1E + 4 932 |
| pointeur court | 2 | 2 | 1 | 0 - FFFFh |
| pointeur long | 4 | 4 | 2 | 0 - FFFFFh |

# Le hash-code démystifié

**Parfois, les programmeurs utilisent des techniques rustiques pour stocker des données : les listes chaînées ont souvent leur faveur.**

**Des langages comme Pascal et C offrent les outils de base pour les réaliser : les pointeurs.**

YVES ROUMAZEILLES

C'est pourquoi les différents manuels de présentation des langages structurés et les ouvrages de base sur la programmation donnent les listes chaînées comme exemple de la gestion de pointeurs. On en trouve donc partout les routines de base.

Malheureusement, de telles structures de données ne sont pas très pratiques pour retrouver un des éléments qui y sont stockés. Il faut parcourir toute la liste avant de savoir si l'élément recherché est présent ou non. On utilise alors des structures auto-adaptatives (1) qui modifient automatiquement l'ordre des éléments de manière à ce que les données récemment utilisées soient en tête de la liste (à peu de choses près, il s'agit de la méthode du cache de données).

Certains des lecteurs auront aussi pensé aux arbres (binaires ou non). Ils ont l'avantage de réduire considérablement le temps de parcours tout en autorisant diverses formes d'optimisation (tri des données selon une clef, équilibrage automatique de l'arbre,...).

Sans doute, l'aspect le plus contraignant de cette technique est sa complexité de codage qui entraîne parfois des erreurs difficiles à corriger (le parcours d'un arbre binaire n'est pas forcément évident à effectuer et à programmer).

En fait, il s'offre une autre solution à ceux qui n'ont pas besoin de trier les éléments : le hash-code. L'avantage majeur de cette technique est la possibilité d'atteindre un élément au premier essai dans la plupart des cas.

## Comment cela marche-t-il ?

Le principe en est très simple (beaucoup plus qu'on ne l'imagine en général). Pour retrouver un élément il faut savoir où il est rangé.

On associe donc à chaque élément stocké une valeur unique calculée à partir de la clef qui sera utilisée pour le rechercher. Cette valeur est utilisée comme indice dans une table. Ainsi, un simple calcul permettra d'avoir l'endroit exact où est stocké l'élément.

Par exemple, dans un compilateur, si l'on souhaite enregistrer les structures de données des éléments lexicaux d'un programme, chaque structure comportera le nom de la variable, son type, sa valeur, et différentes autres informations.

La clef sera le nom de la variable puisque le compilateur utilisera ce nom pour savoir si la variable est déjà déclarée ou pas.

Plusieurs possibilités se présentent pour calculer cette valeur unique (qui prend le nom de hash-code). La fonction de calcul est sélectionnée en fonction des clefs que l'on s'attend à trouver.

Ainsi, dans le programme qui accompagne cet article, il a été adopté une méthode simple mais qui permet de traiter un peu tous les genres de clefs alphabétiques : additionner tous les caractères de la clef, puis ramener cette valeur dans l'intervalle [0.. taille_table-1].

Évidemment, bien d'autres possibilités sont envisageables avec divers avantages et inconvénients : utiliser le nombre de caractères de la clef, additionner le premier caractère avec la longueur de la clef, etc.

Les lecteurs attentifs auront bien sûr noté que, dans le cas général, il est impossible de garantir par avance que deux clefs ne vont pas générer la même valeur.

Une fois que l'emplacement dans la table est occupé, il faut trouver un nouvel emplacement pour le deuxième candidat ou bien les mettre tous les deux "au même endroit".

Parmi les solutions de rechange, l'une consiste à stocker l'élément recherché un peu plus loin dans la table en utilisant un autre calcul pour trouver un nouvel emplacement. En fait, la difficulté apparaît quand la table est très pleine : quelle que soit la méthode employée, il devient très long (puis impossible de trouver une place libre).

Pour éviter cette extrémité catastrophique, la solution que nous avons retenue a l'avantage de l'extensibilité : elle n'est jamais limitée que par la taille de la mémoire disponible et non pas par la taille de la table de hashing choisie à sa création.

Malgré ce que nous avons dit plus haut, nous avons adopté les listes chaînées à cause de leur simplicité de codage. A chaque emplacement de la table est associée une liste. Si un élément doit être ajouté, il est mis en queue de la liste associée à l'emplacement qui a été sélectionné dans la table par la fonction de hashing.

Au moment de la recherche, il faut balayer la liste pour vérifier si l'élément dont on a la clef est présent sur un nœud de la liste. Les inconvénients de ces structures de données sont compensés par le fait qu'elles ne sont utilisées que lorsque les "collisions" interviennent dans la table.

Cela n'intervient que rarement si deux conditions sont remplies :

— la table est suffisamment grande (on considère, en général, qu'elle ne doit jamais être remplie à plus de 40 %) ;

— la fonction de codage est suffisamment adaptée aux clefs couramment utilisées pour assurer une répartition homogène sans trop de collisions (si toutes les clefs ont la même longueur, il est inutile d'utiliser cette information, et il est dangereux de n'utiliser que cela dans le calcul).

## Une implémentation particulière

Afin de rendre les choses plus claires, il était aussi simple de vous proposer un exemple d'implémentation complet. On peut y trouver en langage C une fonction de hashing, aussi bien que les routines de création de la table, d'insertion et de recherche des éléments et, bien sûr, un petit programme de test. Cela permettra au lecteur de faire des essais, mais aussi d'inclure ces petites routines dans son propre code.

La fonction de hashing hash() a déjà été décrite plus haut, mais notons qu'elle est isolée afin de permettre aux lecteurs de la modifier plus facilement pour s'adapter à un besoin précis. Mais attention, ne compliquez pas inutilement cette fonction. De son temps d'exécution dépend directement celui de la fonction hfind(). Les fonctions les plus simples sont généralement les meilleures.

Par exemple, si les clefs doivent être longues, il sera donc préférable de ne totaliser que quelques caractères au lieu de tous les prendre en compte. Une légère augmentation des risques de collisions sera alors aisément corrigée par l'élargissement de la table de hashing.

La routine hcreer() de création de la table alloue la mémoire en appelant la fonction standard malloc(). Elle prend la peine de choisir une taille de table qui soit un nombre entier. Des recherches ont en effet montré que, de manière générale, cela réduit les risques de collisions avec la plupart des fonctions de hashing basées sur un calcul modulo la longueur de la table.

Le nombre sélectionné est simplement le premier nombre entier supérieur au nombre d'éléments suggéré par le paramètre de la routine hcreer(). La routine renvoie un pointeur sur la table ainsi créée. Ce pointeur sera ensuite utilisé comme référence à la table pour l'insertion et la recherche d'éléments.

On notera aussi le fait que l'appel à la routine hcreer() demande un pointeur sur une fonction passé en argument. Cette fonction sera appelée lorsqu'il sera nécessaire de vérifier si une clef décrit bien un élément trouvé dans la table. Elle prend comme premier argument un pointeur sur un élément soumis au test et comme second argument la clef qui est recherchée.

La routine hfind() sert aussi bien à la recherche qu'à l'insertion. Le paramètre def_action indique comment doit être traitée l'erreur consistant à ne pas trouver l'élément dont la clef a été donnée en argument. Si def_action vaut INSERER, alors la fonction fait une insertion dans la table.

Si def_action vaut TROUVER, alors la fonction se contente de dire qu'elle n'a pas trouvé l'élément en renvoyant un pointeur nul.

Cette routine très simple (c'est un avantage des listes chaînées utilisées en cas de collision) utilise un pointeur sur une table de hashing comme premier argument pour permettre de manipuler plusieurs tables en même temps.

Dans le cas du compilateur cité plus haut, il faudra sans doute des tables séparées pour les noms de variables, ceux des types, ceux des fonctions, ceux des procédures,...). Dans celui d'une base de données les index peuvent être multiples.

On trouvera enfin un petit exemple d'utilisation de ces routines qui présente l'avantage d'être un moyen simple de vérifier le bon fonctionnement des routines quand elles ont été recopiées manuellement. On observera alors la manière normale d'utiliser ces routines aussi bien pour l'insertion que la recherche d'éléments dans la table de hashing.

Normalement, les routines fournies ici sont utilisables avec n'importe quel compilateur sur PC, comme sur une autre machine (Macintosh ou Unix par exemple). Les seuls fichiers inclus nécessaires sont ⟨stdio.h⟩ (qui accompagne tous les compilateurs C) et ⟨strings.h⟩ (qui accompagne la plupart des compilateurs C pour déclarer la fonction strcmp() utilisée dans le programme de test).

Les différences entre un compilateur C "à la Kernighan & Ritchie" (comme le Microsoft C v5.1, la plupart des compilateurs sous Unix, ou le Turbo C v2.0) et un compilateur qui suit la proposition de norme ANSI (comme Turbo C++) sont réglées par la présence des sélections de la forme :

```
#ifdef __STDC__
```

que l'on peut trouver dans les fichiers HASH.C et HASH.H.

Cela recouvre principalement le prototypage des fonctions qui doit comporter les arguments quand on utilise un C ANSI, et ne doit rien comporter avec le C de K&R. La différence apparaît dans le fichier HASH.H. Ensuite, on trouve la déclaration par défaut de ELEM qui utilise le type void qui n'existe que dans un compilateur ANSI.

Cette déclaration est importante parce que différente dans le fichier HASH.C et dans le fichier d'utilisation. Dans HASH.C, on ne sait pas à quoi va ressembler le type d'éléments à manipuler. On se contente donc d'une approximation et on utilise des pointeurs sur ceux-ci.

## Les principaux inconvénients du hashing

Ne le cachons pas, nous n'avons pas trouvé la solution miracle qui va régler tous les problèmes. Les performances d'une technique comme le hashing sont effectivement sensiblement meilleures que celles des arbres de données, mais il existe un obstacle majeur à la généralisation de son emploi.

Nous sommes passés rapidement sur le fait qu'il est nécessaire d'avoir une bonne idée de la taille que doit avoir la table. Pour cela, il faut savoir (même approximativement) combien de données seront à y ranger. Cette allocation statique (faite une fois pour toutes, contrairement aux listes et aux arbres) de la table n'est pas toujours possible.

La technique discutée plus haut s'efforce de réduire le problème dans les cas les plus simples, mais n'est pas suffisante si le nombre d'éléments n'est pas prévisible. Si l'on reprend l'exemple du compilateur cité plus haut, il sera préférable d'utiliser une fonction de hashing pour les identificateurs réservés du langage (dont le nombre ne varie pas), et des arbres binaires pour les variables, les types et toutes les constructions dont le nombre varie beaucoup d'un programme à un autre (et de manière imprévisible ou presque).

## Améliorations possibles

Bien sûr, le comportement global de cette bibliothèque peut être sensiblement amélioré dans certains cas particuliers. Par exemple, si les éléments que vous souhaitez ranger dans la table sont connus par avance, il est intéressant de trouver une fonction de hashing dite "optimale" (c'est-à-dire qui ne produit aucune collision).

Il sera ainsi possible de modifier le code de hfind() pour retirer la gestion des listes chaînées si l'on peut garantir par avance l'absence de collisions. C'est le cas pour les identificateurs réservés d'un langage qui sont tous connus.

Toute la difficulté réside alors dans le choix de la bonne fonction de hashing qui dépend complètement du problème posé (les identificateurs réservés de Pascal et C exigeront une fonction différente).

Afin de vous aider dans la recherche d'une solution à un problème particulier, nous vous signalons qu'il est souhaitable de faire jouer la taille de la table en plus des paramètres de la fonction de codage.

Cette technique de hashing peut aussi être utilisée pour manipuler des objets dont la clef n'est pas une chaîne de caractères. On peut penser à des clefs représentées par des nombres réels en notation à virgule flottante, des couples de nombres (les coordonnées d'un point sur un écran), une structure complète (pourquoi pas ?).

Ensuite, il est sans doute souhaitable de pouvoir retirer des objets de la table. Une routine qui recherche un élément en vue de sa destruction est facilement envisageable. Elle aurait beoucoup de points communs avec la routine hfind().

Enfin, pourquoi un lecteur n'essaierait-il pas de programmer cette technique dans un langage orienté-objet ? La table de stockage est alors l'objet central (un dépôt, un garde-meuble, etc.) que l'on crée, auquel on peut appliquer les méthodes "ranger un élément", "retrouver un élément", et qu'il faudra aussi savoir détruire quand on n'en a plus l'usage (cette opération est partiellement contenue dans la gestion d'erreurs de hcreer()).

Le portage du code joint à cet article sera assez facile dans presque tous les langages orientés-objet (mais encore plus en C++ ou en Objective-C).

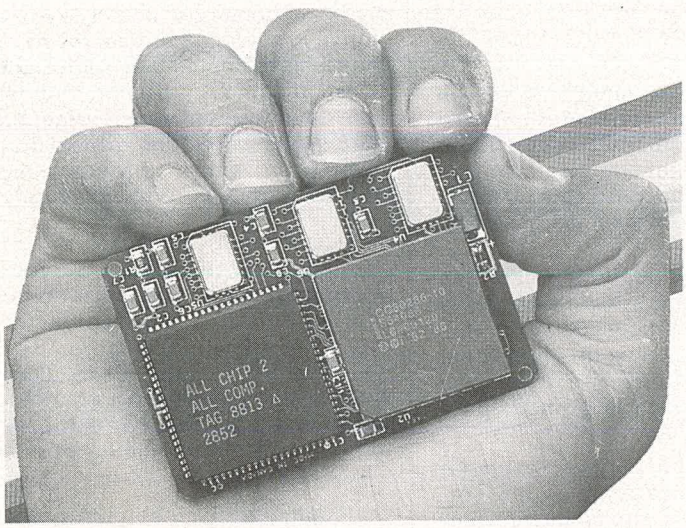
(1) Voir à ce sujet, "Self-Adjusting Data Structures" par Andrew M. Liao (Dr. Dobb's Journal, February 1990).


Nous facturons
1 950 FHT

FLI REMBOURSE LA TVA SUR LA CHARGECARD

Vous payez
1 950 FTTC

Vous estimez que DOS est toujours dans la course. Profitez de la fiabilité et de l'expérience ALL Computers en utilisant intelligemment la mémoire étendue de vos 286 :




Ayez 640 KO pour travailler, même si les machines ne l'ont pas prévu

Mieux !, étendez cet espace à 704 KO (MDA, HERCULES) voire même 736 KO (CGA)

Encore plus fort ! recupérez 192 KO et plus au dessus de la mémoire vidéo pour les couches de base du DOS et les programmes TSR (réseaux, émulations, souris, clavier, ...)

Attribuez de la mémoire EMS 4.0 à LOTUS 123, CLIPPER, DESQview, ...

Augmentez notablement les capacités de WINDOWS 3

CHARGECARD se rit des BUS ISA ou MCA
Elle existe aussi pour les portables TOSHIBA et COMPAQ

Vous hésitez à ouvrir votre machine ? un spécialiste FLI peut se charger de l'installation.

Vous cherchez un vrai support technique ? inscrivez le numéro de téléphone de FLI dans votre agenda.

Vous ne comprenez pas l'Anglais ? FLI vous propose en option une documentation en Français.

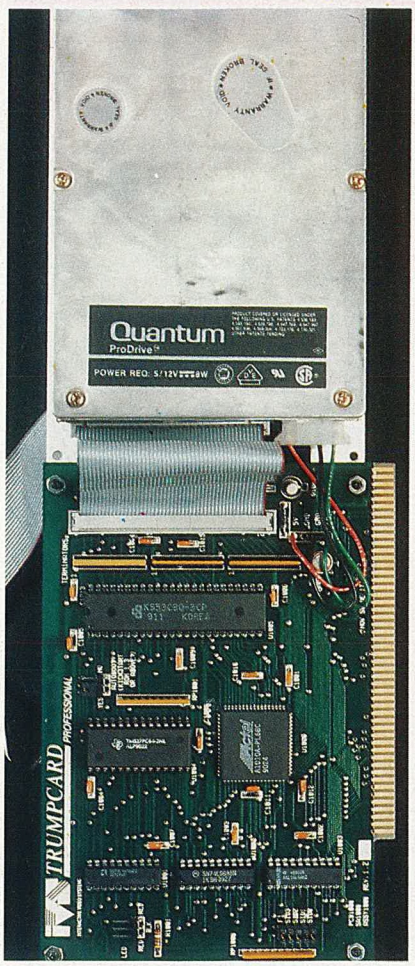
La dernière version du logiciel ? Elle vous attend, gratuitement, sur un serveur télématique.

Vous n'avez pas de mémoire étendue ? le meilleur rapport qualité/prix du marché se trouve chez FLI !

Vous n'avez pas de machine 286 ? ALL Computers a pensé également aux machines 8088 et 80386/80486 !

TEL : (1) 60 26 72 00 — FLI Place du Canal, 77410 CLAYE SOUILLY — FAX : (1) 60 26 28 10

```
/*
 *      SCCS Id: "@(#)primes.h  (C) Yves Roumazeilles    11/05/90";
 */

/* Les nombres premiers inférieurs à 32768 */
int     primes[] = {
/*      1,      2,      */
        3,      5,      7,      11,     13,     17,     19,     23,
        29,     31,     37,     41,     43,     47,     53,     59,
        61,     67,     71,     73,     79,     83,     89,     97,
        101,    103,    107,    109,    113,    127,    131,    137,
        139,    149,    151,    157,    163,    167,    173,    179,



/* ... */



        32321,  32323,  32327,  32341,  32353,  32359,  32363,  32369,
        32371,  32377,  32381,  32401,  32411,  32413,  32423,  32429,
        32441,  32443,  32467,  32479,  32491,  32497,  32503,  32507,
        32531,  32533,  32537,  32561,  32563,  32569,  32573,  32579,
        32587,  32603,  32609,  32611,  32621,  32633,  32647,  32653,
        32687,  32693,  32707,  32713,  32717,  32719,  32749
};

#define NPRIMETAB       (sizeof(primes)/sizeof(primes[0]))
```





## *LE VRAI SEMINAIRE CLIPPER 5.0 DE NANTUCKET POUR DES DEVELOPPEURS PAR DES SUPER DEVELOPPEURS*

**Pour vous perfectionner ou débuter en 5.0**
**2 opportunités : le 4 et 5 Juin niveau débutant 5.0**
**Le 6 et 7 Juin niveau intermédiaire et avancé**

INSCRIVEZ-VOUS DES MAINTENANT.

Avec pédagogie, directement orienté vers la productivité, la programmation efficace, des ''CLIPPER gourous'', tels GREG LIEF, vous enseigneront 5.0.
Des notes dactylographiées, des disquettes de programmes, quantité de routines vous seront remises à l'issue du séminaire.

Séminaire en Français et en Anglais avec traduction simultanée.

Participation : 3 600 Frs HT la session (inscription après le 15 Mai 4 100 Frs HT).

ILS ANIMENT TOUTES LES CONFERENCES NANTUCKET AUX USA

**Lieu : Centre National d'Etudes Spatiales**
**2 Place Maurice Quentin - 75002 Paris**
**(Métro : Les Halles)**

— Repas : midi inclus.
— Réduction de 35 % sur vol AIR INTER.
— Formation Continue : Convention n° 93060180606.

LDQS - 5 rue Eugène-Emanuel - 06000 Nice

Tél. : 93 16 09 12 -
Fax : 93 88 08 90

COUPON DE PARTICIPATION
LDQS - 5 rue Eugène-Emanuel - 06000 Nice
Tél. : 93 16 09 12 - Fax : 93 88 08 90

Session le 4 et 5 Juin ☐ Session le 6 et 7 Juin ☐

Nom ..... Société .....
Adresse ..... Tél. .....
Code Postal ..... Ville .....

```
#if     !defined( lint )
static  char *sccsid = "@(#)hash.c      (C) Yves Roumazeilles 11/01/90";
#endif

/*
 *
 */

#ifdef  __STDC__
typedef void    ELEM;
#else
typedef int     ELEM;
#endif

#include"hash.h"
#include"primes.h"

/*
 *  Construction de la table.
 */

HASHTBL*
hcreer( nb, fcmp )
int     nb;
#ifdef  __STDC__
int     (*fcmp)(ELEM*,char*);
#else
int     (*fcmp)();
#endif
{
        HASHTBL     *htp;
        int         taille;
        int         i;

        htp = (HASHTBL*)malloc( sizeof( HASHTBL ) );
        if( htp == NULL )
                return( NULL );

        /* Recherche de la taille optimale */
        if( nb > primes[NPRIMETAB-1] )
                taille = nb;
        else
                for( taille=1, i=0; taille<nb && i<NPRIMETAB; i++ )
                        taille = primes[i];

        htp->h_nb_elem = taille;
        /* Allocation de la table proprement-dite */
        if( (htp->h_table = (NOEUD**)malloc(taille*sizeof(NOEUD*))) == NULL ) {
                free( htp );
                return( NULL );
        }
        for( i=0; i<taille; i++ )
                htp->h_table[i] = NULL;

        htp->h_fcmp = fcmp;
        return( htp );
}
```

```
/*
 *  Chercher l'entrée stockée dans la table pointée par htp et
 *  correspondant à la clef key.
 *
 *  Si la clef n'est pas trouvée dans la table, l'argument def_action
 *  est utilisé pour choisir l'action suivante:
 *      INSERER     insérer l'élément pointé par elemp,
 *                  le renvoyer en résultat.
 *      TROUVER     ignorer, renvoyer NULL.
 *
 *  Dans tous les cas d'erreurs, la fonction renvoie NULL.
 *
 *  Utilise la fonction de hash-coding hash().
 */

ELEM*
hfind( htp, elemp, key, def_action )
HASHTBL     *htp;
ELEM    *elemp;
char    *key;
int     def_action;
{
        int     hvalue;
        NOEUD**noeudpp;

        /* Trouver l'entrée correspondante dans la table */
        hvalue = hash( htp, key );
        noeudpp = &(htp->h_table[hvalue]);

        /* Chercher dans la liste chaînée associée à l'entrée */
        while( *noeudpp != NULL ) {
                if( (*htp->h_fcmp)( (*noeudpp)->n_elemp, key ) == 0 ) {
                        /* trouvé */
                        return( (*noeudpp)->n_elemp );
                }
                else {
                        /* pas trouvé, continuer le long de la liste chaînée */
                        noeudpp = &((*noeudpp)->n_ptr);
                }
        }

        /* On ne l'a trouvé nulle part ... */
        switch( def_action ) {
        case INSERER:
                *noeudpp = (NOEUD*)malloc( sizeof(NOEUD) );
                (*noeudpp)->n_ptr       = NULL;
                (*noeudpp)->n_elemp     = elemp;
                return( elemp );
        case TROUVER:
        default:
                return( NULL );
        }
}


/*
 *  Calcule le hash-code de la chaîne de caractères text dans la table
 *  pointée par htp.
 *
 *  La formule utilisée est une simple somme de tous les caractères de
 *  la chaîne (la somme est corrigée en additionnant une fois
 *  supplémentaire le premier caractère).
 */

int
hash( htp, text )
HASHTBL             *htp;
register char       *text;
{
        register int        value = 0;

        while( *text )
                value += *text++;
        return( value % htp->h_nb_elem );
}
```

# Si vous êtes déjà un utilisateur de CLIPPER de Nantucket, vous avez besoin du

## Générateur de programme
## ATELIER

**Il fait pour vous :**

☐ la génération du menu principal,
☐ la génération de sous-menus,
☐ la génération des programmes de saisie,
☐ la génération des programmes d'impression,
☐ la gestion de vos sources,
☐ le lancement du compilateur...

## ATELIER

☐ incorpore votre copyright et une calculette dans vos programmes
☐ génère le code fastidieux (90 %)
☐ et vous laisse faire la partie intéressante de la programmation.

## ATELIER

☐ est livré avec les sources commentées, ainsi qu'une documentation imprimable.

La version compilée n'est pas livrée sur la disquette car ATELIER est destiné aux utilisateurs de CLIPPER et nécessite ce compilateur.

***Les commentaires sur ce programme ont été publiés dans La Revue de l'Utilisateur de l'IBM PC (numéro 59, Janvier-Février 1990), et la suite de la série comportera 6 articles à compter du numéro 61, Avril 1990.***

## ATELIER

**est disponible auprès de La Revue de l'Utilsateur de l'IBM PC au prix de 450 F, sur disquette 5''1/4 ou 3''1/2.**

---

**COUPON**
**à adresser à La Revue de l'Utilisateur de l'IBM PC**
**2, rue d'Amsterdam**
**75009 Paris**
**Tél. : 47.64.07.57. - 42.67.93.80**

**Nom** ..... **Prénom** .....
**Adresse** .....
.....
**Code Postal** ..... **Ville** .....
**Téléphone** .....
**Veuillez m'envoyer** ..... **logiciel (s) ATELIER au prix unitaire de 450 F (TTC).**
**Veuillez trouver ci-joint un chèque de** ..... **F en règlement à Softin Communications.**

```
/*
*       SCCS Id: "@(#)hash.h    (C) Yves Roumazeilles    11/01/90";
*/

#include<stdio.h>       /* pour définir NULL */

#define TROUVER     0
#define INSERER     1

typedef struct noeud {
        struct noeud    *n_ptr;
        ELEM            *n_elemp;
} NOEUD;

typedef struct hashtbl {
        NOEUD *                 *h_table;
        int                     (*h_fcmp)();
        int                     h_nb_elem;
} HASHTBL;


/* prototypes des fonctions */
#ifdef  __STDC__
HASHTBL* hcreer(int,int(*)(ELEM*,char*));
ELEM*   hfind(HASHTBL*,ELEM*,char*,int);
int     hash(HASHTBL*,char*);
#else
HASHTBL* hcreer();
ELEM*   hfind();
int     hash();
#endif
```

```
#if     !defined( lint )
static  char *sccsid = "@(#)main.c      (C) Yves Roumazeilles,11/01/90";
#endif

/*
 *  Fichier de test pour HASH.C
 */

#include<stdio.h>
#include<string.h>


/*
 *  L'entité de stockage.
 *
 *  Attention: cette déclaration doit apparaître avant l'inclusion
 *  du fichier hash.h.
 */
typedef struct elem {
        char    *e_name;
        int     e_value;
} ELEM;


#include"hash.h"

/* Quelques éléments pour le test */
ELEM    elements[] = {
        { "test0", 0 },
        { "test1", 10 },
        { "test2", 20 },
        { "test3", 30 },
        { "test4", 40 },
        { "test5", 50 },
        { "test6", 60 },
        { "test7", 70 },
        { "test8", 80 }
};


int
elem_cmp( ep, clef )
ELEM    *ep;
char    *clef;
{
        return( strcmp( ep->e_name, clef ) );
}



main()
{
        HASHTBL     *htp;
        ELEM    *elemp;

        /* La création de la table */
```

```
htp = hcreer( 5, elem_cmp );

/* Les insertions */
hfind( htp, &elements[0], elements[0].e_name, INSERER );
hfind( htp, &elements[1], elements[1].e_name, INSERER );
hfind( htp, &elements[2], elements[2].e_name, INSERER );
hfind( htp, &elements[3], elements[3].e_name, INSERER );

hfind( htp, &elements[6], elements[6].e_name, INSERER );
hfind( htp, &elements[7], elements[7].e_name, INSERER );

/* Les recherches */
elemp=hfind( htp, NULL, "test1", TROUVER );
if( elemp == NULL )
        printf( "introuvable\n" );
else
        printf( "%s = %d\n", elemp->e_name, elemp->e_value );

elemp=hfind( htp, NULL, "test5", TROUVER );
if( elemp == NULL )
        printf( "introuvable\n" );
else
        printf( "%s = %d\n", elemp->e_name, elemp->e_value );

elemp=hfind( htp, NULL, "test6", TROUVER );
if( elemp == NULL )
        printf( "introuvable\n" );
else
        printf( "%s = %d\n", elemp->e_name, elemp->e_value );
```

NOUVEAU

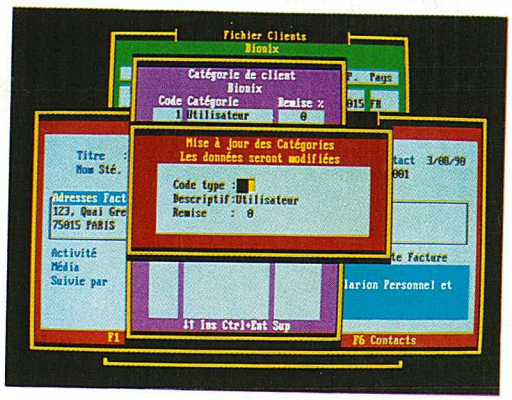
# CLARION Personnel N° 1 aux États-Unis en version française

## Le Générateur d'applications intégral

***Le moyen simple pour réaliser des applications de gestion de toute nature***

**Créez vos applications,
de A à Z en quelques minutes
c'est ce que CLARION Personnel
le générateur le plus primé aux U.S.A.
réalise pour vous.**

**Fichiers, index, écran multifenêtrés :
menus, tableaux déroulants, masque de saisie, états ou mailings, champs calculés, conditionnels, totaux...
pour des applications SGBD Relationnel de qualité distribuables SANS REDEVANCE.**




**Prix tarif 2 390,00 Francs HT**

## PRIX DE LANCEMENT 990,00 * Frs. HT

**Spécifications techniques :**

**Taille des fichiers :** 252 fichiers par application, 251 clés par fichier, 255 champs par enregistrement.
**Longueur enregistrement jusqu'à 64 Ko.**
**Mode automatique :** Ilimitée.
Crée une application totale en quelques secondes (menus, tableaux déroulants, masques de saisie, états) à partir de la simple définition du fichier.
**Aide à l'utilisation :** Intégrée (plus de 400 écrans).
**Conversion de fichiers :** Import/export avec autre base de données et tableur.

## Bon de commande

**Nom** .....
**Société** .....
**Fonction** .....
**Adresse** .....
**Code Postal** ..... **Ville** .....
**Tél** .....

**Je désire recevoir CLARION Personnel pour le prix de lancement de 990,00 Francs HT plus la TVA et ma participation aux frais d'envoi soit 1 249,14 Francs TTC.**

**Veuillez trouver ci-joint mon chèque de règlement à l'ordre de Bionix.**

***Signature*.....**

*** Offre valable jusqu'au 31 Décembre 1990.**


**BIONIX**
**34, rue de la Saussière**
**92100 Boulogne - FRANCE**
**Tél. : (33)-(1) 46 05 74 89**
**Fax. : (33)-(1) 46 05 01 77**

**Pour la Suisse**
**IHS DISTRIBUTION S.A.**
**En Burdon A 16**
**1052 Le Mont / Lausanne (SUISSE)**
**Tél. : (41)-(021) 652 34 36**
**Fax. : (41)-(021) 652 34 52**

# GUERRE DES PRIX ? DYNAMIT COMPUTER PRESENT

## LA MEILLEURE OFFRE DU MARCHE FRANÇAIS !

## AYEZ PITIE DE VOTRE POCHE MAIS SURTOUT DE VOS YEUX !

DYNAMIT
COMPUTER

**LE DS/3 COMPREND POUR 6 990,00 F TTC (5 893,76 F HT) !**

a) boîtier compact, alimentation 200W UL/FCC (norme USA), carte mère 80286-12 Indice Norton 15,8, 1MoRAM, 1 lecteur 3''1/2 1,44 Mo, disque dur 40 Mo Maxtor 28 ms, carte VGA 16 bits, carte contrôleur, lecteur et disque dur, série (2), parallèle 1, clavier 1021 AZERTY, DR-DOS 5.0 + manuel français.

b) moniteur VGA couleur

c) émulation Minitel par connection « Enter »

d) souris NEOS (Japon, ouf, oui pas Taïwan)

e) une garantie sur site d'un an !

*(photo non contractuelle)*

**PRIX LE PLUS BAS DE FRANCE : 6 990,00 F TTC (5 893,76 F HT)**
**PRIX AVEC MONITEUR SONY PITCH 0,25 (POUR VOS YEUX !) : 7 690,00 F TTC**

**IMPRIMANTES**
OKI 400L : LA PREMIERE IMPRIMANTE LASER A 6 950,00 F TTC !
Caractéristiques : 4 pages/minutes, 512 Ko, idéal traitement de texte !

Citizen 1200+ : 1 250 F TTC
Citizen Swift 24 : 2 950 F TTC
Star LC 20 : 1 990 F TTC
LC200 couleur : 2 990 F TTC
**Canon BJ10e : 2 790 F TTC**
Spécial BJ10e+BAC F/F+cable : 2 862,56 F HT (3 395 F TTC)

*(photo non contractuelle)*

**Lecteurs et disques durs**
5''1/4 1,2 Mo : 490 F TTC
3''1/2 1,44 Mo : 490 F TTC
Berceau 5''1/4 3''1/2 : 70 F TTC
12 Mo MFM : 695 F TTC
40 Mo AT BUS : 1 950 F TTC
80 Mo AT BUS : 3 990 F TTC
110 Mo AT BUS : 4 790 F TTC
STREAMER 40/60 Mo : 2 590 F TTC

**Moniteurs**
VGA couleur : 2 490 F TTC
NEC 2A : 4 290 F TTC
NEC 3D : 5 590 F TTC
NEC 4D : 10 250 F TTC
NEC 5D : 20 900 F TTC

**Compétences Réseau Novell, installation et formation, compétences télématique, installation de serveur sous UNIX**
**Contactez-nous pour une liste complète de nos produits :**
Ordinateurs 386SX-16/20,80386-25, 80386-33, 80486-33, made in USA.
Assemblée dans notre usine de 1 000 m² et bénéficiant d'un vrai burn-in
Erreurs et omissions exceptées

**Je désire avoir plus d'information sur l'article suivant :**

.....
Nom : .....
Prénom : .....
Profession : .....
Adresse : .....
.....

**DYNAMIT COMPUTER**
77, rue de Maubeuge - 75010 PARIS
Tél. : 42 82 17 09
Fax : 42 82 17 25

MM2 COMMUNICATION - 45 94 95 95